5-16-2011

MapSnap System to Perform Vector-to-Raster Fusion

Boris Kovalerchuk

Peter Doucette

Gamal Seedahmed

Jerry Tagestad

Sergei Kovalerchuk

See next page for additional authors

Authors
Boris Kovalerchuk, Peter Doucette, Gamal Seedahmed, Jerry Tagestad, Sergei Kovalerchuk, and Brian Graff

MapSnap System to Perform Vector-to-Raster Fusion

Boris Kovalerchuk[ab*], Peter Doucette[c], Gamal Seedahmed[d], Jerry Tagestad[e], Sergei Kovalerchuk[b], Brian Graff[f]

[a]BKF Systems, 8241 S. 123 St., Seattle, WA, 98178,
[b] Central Washington University, Ellensburg, WA 98926-7520,
[c] Integrity Applications Inc., 5160 Parkstone Dr., Chantilly, VA 20151,
[d]NG4, 10780 N. Preserve Way 308, Miramar, FL, 33025,
[e]Battelle Pacific Northwest Division, Richland, WA 99352,
[f]US Army ERDC Geospatial Research and Engineering Division, Telegraph Rd Bldg 2592, Alexandria, VA 22315

ABSTRACT

As the availability of geospatial data increases, there is a growing need to match these data sets together. However, since these data sets often vary in their origins and spatial accuracy, they frequently do not correspond well to each other, which create multiple problems. To accurately align with imagery, analysts currently either: 1) manually move the vectors, 2) perform a labor-intensive spatial registration of vectors to imagery, 3) move imagery to vectors, or 4) redigitize the vectors from scratch and transfer the attributes. All of these are time consuming and labor-intensive operations. Automated matching and fusing vector datasets has been a subject of research for years and strides are being made. However, much less has been done with matching or fusing vector and raster data. While there are initial forays into this research area, the approaches are not robust. The objective of this work is to design and build robust software called MapSnap to conflate vector and image data in an automated/semi-automated manner. This paper reports the status of the MapSnap project that includes: (i) the overall algorithmic approach and system architecture, (ii) a tiling approach to deal with large datasets to tune MapSnap parameters, (iii) time comparison of MapSnap with redigitizing the vectors from scratch and transfer the attributes, and (iv) accuracy comparison of MapSnap with manual move of vectors. The paper concludes with the discussion of future work including addressing the general problem of continuous and rapid updating vector data and fusion vector data with other data.

Keywords; Geospatial Science and Engineering Applications, image, vector data, registration, conflation, alignment, vector data, automated feature extraction.

1. INTRODUCTION

As the availability increases of geospatial data, there is a growing need to match these data sets together. However, since these data sets often vary in their origins and spatial accuracy, they frequently do not correspond well to each other. This lack of spatial correspondence creates multiple problems. To get the existing vector data to accurately align with imagery, analysts currently either have to: 1) manually move the vectors, 2) perform a labor-intensive spatial registration of vectors to imagery, 3) move imagery to vectors (rubbersheet), or 4) redigitize the vectors from scratch and transfer the attributes. All of these are time-consuming operations and labor-intensive.

Automated matching and fusing vector datasets together has been a subject of research for many years and strides are being made. This type of fusion is called vector-to-vector conflation. However, much less research has been done with matching or fusing *between* vector and raster data - i.e. vector-to-image conflation. While there are initial forays into this research area, the approaches are not robust nor commercially available. Specifically, automated/semi-automated techniques are needed to conflate vector and image data.

[*]bkf1@BKFSystems.com; phone/fax 509 857-2500;

The objective of the study is to design and build software that will conflate vector and image data in an automated/semi-automated manner. While the goal is total automation, it is recognized that this may not be possible. However, a semi-automated approach must be able to reduce significantly the manual steps and time necessary to align the vector and imagery data. The software must minimally be able to perform the following: 1) allow conflation to be performed in either direction (vector-to-image or image-to-vector; however, vector-to-image is the priority), 2) work with road vectors, 3) work with feature vectors and imagery where the spatial displacement is non-systematic (i.e. a simple translation will not suffice), 4) work with both panchromatic and multi-spectral imagery, 5) work with high-resolution imagery (4 meters and better), 6) move all corresponding vector data layers even if matching and conflation are only based on one vector data set (for example, if road vectors are matched and conflated to imagery, move all corresponding ancillary vector layers along with the roads), and 7) make analyst approved changes persistent (i.e. conflation is not just for real-time display).

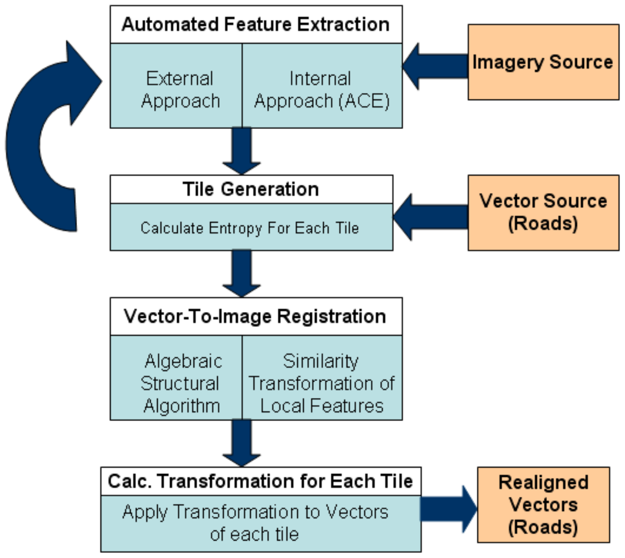

Automated Feature Extraction
External Approach
Internal Approach (ACE)
Imagery Source
Tile Generation
Calculate Entropy For Each Tile
Vector Source (Roads)
Vector-To-Image Registration
Algebraic Structural Algorithm
Similarity Transformation of Local Features
Calc. Transformation for Each Tile
Apply Transformation to Vectors of each tile
Realigned Vectors (Roads)

Figure 1. Flowchart for Vector-to-Image Conflation.

The basic approach that we developed in the MapSnap system for vector-to-image conflation is illustrated in Figure 1. High-resolution imagery is ingested into the software. An initial set of roads is extracted and vectorized through automated feature extraction (AFE). The original imagery, the extracted vectors, and the original road vectors are tiled. Tiling is necessary due to the nonlinear displacement of the original road vectors from the imagery. An entropy measure is calculated for each tile based on the roads extracted. This entropy measure is used to assess the quality of the feature extraction for that tile – have enough high quality vectors been extracted for that tile to warrant future processing? If not, more vectors need to be extracted. Thus, the loop backs to feature extraction. If the tiles are acceptable, then for each tile the original vectors are matched against the extracted vectors using two complementary approaches [4-6, 10]: (1) Algebraic Structural Approach (ASA) algorithm, and (2) Similarity Transformation of Local Features (STLF) algorithm. After the matches are calculated for each tile, a transformation equation is calculated and all of the original road vectors for that tile are moved accordingly. The result is an output file containing the realigned vectors from the original vector source. (Note: this is a very broad description of the process flow: the actual steps are more complicated and detailed).

The MapSnap system has several configurations. The configuration that differs from the basic one captured in Figure1 has an additional block of global transform done by STLF before generation and tiling. For some data, the global transform is sufficient. All configurations offer a user an option to edit extracted file interactively and to rerun MapSnap on the edited data. So far, in experiments data aligning using MapSnap was 4 times faster than re-digitalization from scratch.

The remainder of the paper is organized as follows: Section 2 provides a brief background on relations between accuracy of feature extraction and alignment of vector data with imagery; Section 3 overviews MapSnap systems; Section 4 describes a tiling approach; Section 5 describes experiments with MapSnap and different feature extraction strategies to analyze time and accuracy of alignment, and Section 6 offers some conclusions and future research directions.

2. BACKGROUND

In batch methods, the algorithm attempts to extract all instances of road features throughout an entire scene in one computational session without user inputs. Batch methods often generate errors that must be found through a manual search of an entire scene, which is a significant additional overhead to the overall editing time.

Batch methods can be divided into two categories of 1) *scratch* and 2) *update*. Scratch methods are not based upon using pre-existing vector data. Rather, they assume a 'new' image-based extraction from scratch. Examples for road extraction scratch methods include [1,2]. On the other hand, update methods attempt to leverage *existing vector data* to constrain the search space for image features. Update methods can exploit the fact that over time, the quality, resolution, density, and global extent of geospatial vector data steadily increases. Therefore, a practical assumption is that over time AFE methods will incorporate updating techniques in favor of the brute force techniques used in scratch methods. Examples of update methods for road extraction include [3-7].

Relation between AFE accuracy and alignment. Traditionally it is assumed that the major challenge in designing of the matching algorithms is to find best possible matches to extracted roads. However, alignment correctness is sensitivity to accuracy of automated feature extraction (AFE). Consider a situation where vertical extracted lines are correct roads but horizontal ones are not. Any algorithm that will use these incorrect horizontal lines to align raster to vector will produce incorrect alignment. Thus if even the algorithm will find the best match it does mean that it will align correctly the vector data with imagery.

In fact, the challenge is two-fold: (1) to find a best match and (2) filter out wrong best matches produced by automated feature extraction. The conflation task will become very difficult if we have more wrong extracts than correct ones, $S/N<0.5$. In this case, the noise dominates the signal with low signal noise ratio (SNR) and adding correct inputs to improve this SNR is required. Therefore, a semi-automatic system is a feasible solution, where a user adds some correct roads and/or eliminating incorrect ones.

Figure 2(a) shows another challenge that is the difference between local and global transforms (shifts). Here, while the global transform (green) "globally" correct it is not "locally" correct. A local transform (blue) that uses data only from a specific tile is more accurate. Figure 3(b) shows the opposite case where the global transform's alignment is more accurate than the local one. This means that the algorithm did not capture local specifics and/or local data have higher noise to signal ratio. Figures 3(c) and 3(d) show situations where local and global transforms produce identical results. Note that all cases (a)-(d) are parts of the same large image and the global transform is the same for all of them.

Commercially available AFE methods are imperfect to varying degrees, depending upon a number of factors such as image quality, resolution, scene complexity, etc. The expectation is that some level of inaccuracy will occur in the process. Scratch AFE methods assume no prior information exists, and image features are to be extracted from 'scratch'. However, existing vector data can be leveraged into scratch AFE methods, effectively turning it into a *vector updating method*.

Figure 3 shows scratch AFE output in blue for road centerlines overlaid on test scene 1 zoomed. As expected, errors of commission and omission result, since the image contains relatively complex urban scene content. The issue to be determined is whether a scratch AFE method can provide sufficient levels of accurate output to leverage into a registration process that attempts to align existing vectors with AFE vectors. Once the alignment is achieved, the AFE vectors can be selectively discarded in favor of the newly aligned existing vectors.

The presented examples illustrate a fundamental tradeoff between

(1) minimizing *time for manual extracting and matching features* (by extracting and matching fewer numbers of features manually or avoiding manual work at all) and

(2) minimizing *time for manual inspection* to ensure that the match is accurate enough.

Any approach that minimizes time (1) will pay for it by increasing time (2) to ensure the correctness of alignment. The goal is to minimize the total sum of manual work time for feature extraction, matching and inspection:

$$T_{total} = T_{extract} + T_{match} + T_{inspect}$$

We address this tradeoff problem by exploring and using semi-automatic and semi-manual approach.

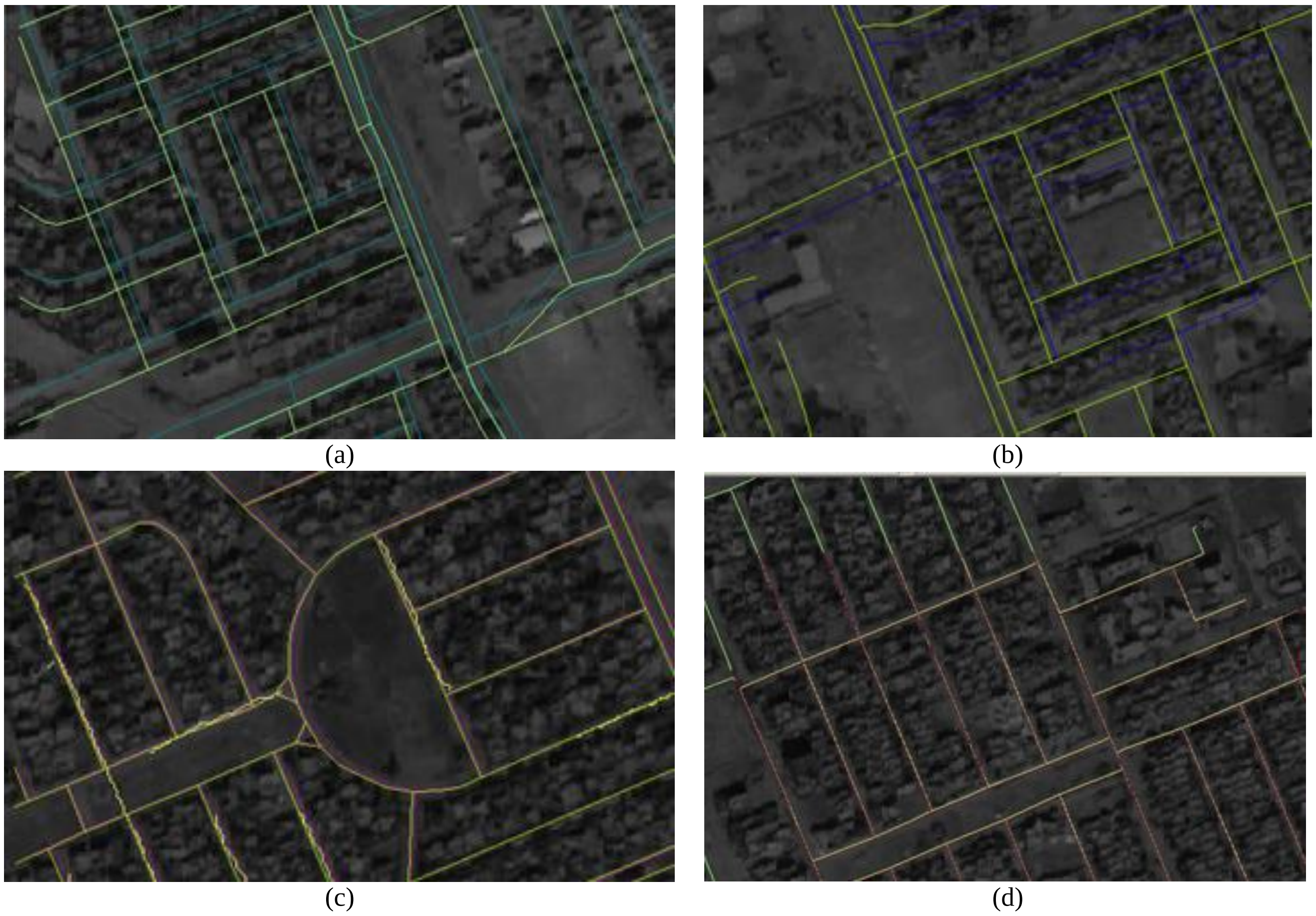


Figure 2. (a), (b) the difference between local transform (blue) and global transforms (green).
(c), (d) consistent local and global transforms.

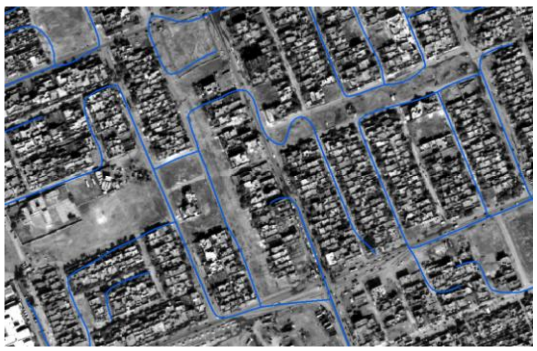
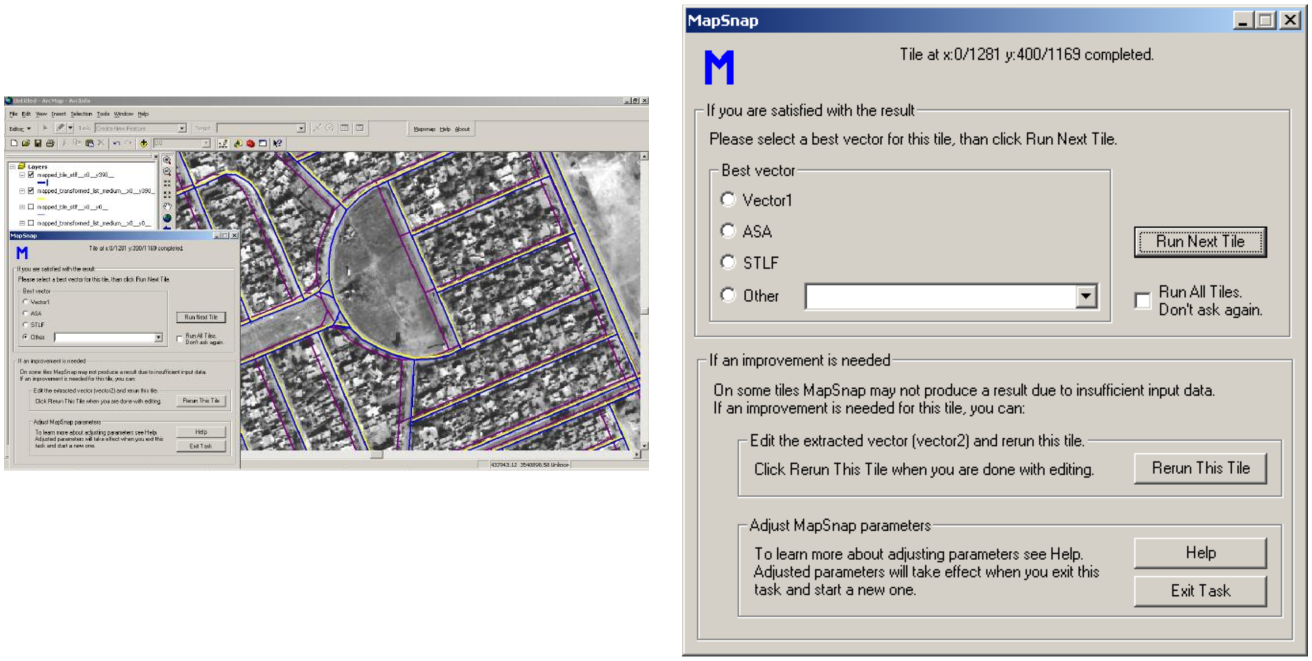


Figure 3. Scratch AFE output (blue) performed on test scene 1.

3. MAPSNAP OVERVIEW

The *MapSnap* system allows a user to select interactively the best result on each tile between produced by ASA and STLF algorithms implemented in the system as well as original vector 1 or even another vector that has been created outside of MapSnap. This GUI also allows a user calling the ArcGIS editor to correct vector 2 and rerun MapSnap on the corrected vector 2. Figure 4 shows MapSnap user interface. The current implementation preserves the original topology of the data. The system creates a new vector 3 where all generated tiles are connected.


MapSnap

Tile at x:0/1281 y:400/1169 completed.

If you are satisfied with the result

Please select a best vector for this tile, than click Run Next Tile.

Best vector

- Vector1
- ASA
- STLF
- Other

Run All Tiles. Don't ask again.

If an improvement is needed

On some tiles MapSnap may not produce a result due to insufficient input data.
If an improvement is needed for this tile, you can:

Edit the extracted vector (vector2) and rerun this tile.

Click Rerun This Tile when you are done with editing.

Adjust MapSnap parameters

To learn more about adjusting parameters see Help.
Adjusted parameters will take effect when you exit this task and start a new one.

Figure 4. MapSnap system interface at ArcGIS plug-in.

At the current state of the art typically, results of automated feature extraction (AFE) are incomplete and incorrect in multiple extracted roads. What are the reasons for using such relatively low quality AFE results for alignment of existing road vector data with imagery? We assume that not all extracts are wrong, but some extracts are correct. However, we do not know which ones are correct before users' inspection that requires significant $T_{inspect}$ time. Potentially these data are better than having nothing especially after time $T_{inspect}$ spent.

The second reason is the *Generalization Assumption* (GA) that the transform F, F(A)=E, that correctly aligns the road A (from incorrect vector 1) with the line E (extracted by AFE method), will also align correctly road B from vector 1 when road B was not extracted by this AFE method. Otherwise, we have no basis to transform B with F, F(B), and need to limit alignment only to roads that are correctly extracted. This *minimalist solution* is rigorous but has low practical value if, say, only 5% or roads are correctly extracted and inspected on correctness by the user. In this case transform F will only align 5% of the roads in vector 1. In contrast, the generalization assumption has a practical value (transform F can align the whole road network with imagery), but this assumption can be false for the data at hand.

How can we test the generalization assumption? How often is GA true? If GE assumption is more specific, e.g., transform F is global linear shift applicable for the entire vector 1 then we can divide data to training and testing parts, discover F in the training part and test F on the testing part. Such test itself assumes that (1) roads are extracted correctly and (2) extracted roads are representative for the all roads in vector 1. As we already discusses, we cannot take (1) as granted. If only 30% of roads are extracted correctly, but 70% of roads are extracted incorrectly then likely the correct F will not be discovered and confirmed on these data.

This leads to the necessity to change the GA to a *local generalization assumption* (LGA): a single transform F_i exists locally at the level of the individual tile S_i. Thus, to transform the entire vector 1 we need a set of transforms $\{F_i\}$. These transforms should be coordinated at the edges of tiles to preserve geometry and topology of the road network. Now the size of the tile becomes the critical parameter of the alignment process. MapSnap has a mechanism to adjust tile size using several methods including entropy estimates.

Now consider a case when individual road A has its own transform F_A that differs from the transform F that is common for all other roads. To find this transform we need (1) to ensure that the matching line E is extracted from the imagery, (2) single out this matching line E from competing lines, and (3) build transform F_A , $F_A(A)=E$. While ASA algorithm has this capability, we left it in MapSnap for a sophisticated user because the probability the AFE will extract such unique road correctly and completely is low. For the most of the users, a more practical way is correcting such road manually (semi-manually). In summary, today MapSnap is most useful for the data that have either: (1) a single linear shift (STLF), (2) a set of different linear shifts, one shift per tile (STLF and ASA), and (3) a set of different affine transforms, one transform per tile (ASA). MapSnap is also useful for data that do not satisfy exactly (1)-(3) but close enough to (1)-(3) that is with errors within acceptable limits for the user.

4. TILING

Tiling has several reasons such as that controlling memory and computing time limitations, tuning MapSnap parameters to fit different image resolutions, road width etc, and making the user inspection and correction of MapSnap results of manageable size. Data tiling helps to solve the alignment task for large data sets that have different transforms at different tiles. While the tiling helps to solve these problems, it creates its own problems. The tiling produces separate tiles that need to be linked together to produce a single vector file. This file is called vector3 in the MapSnap System. The challenges here are:

- Preserving the road network connectivity/topology
- Preserving straight road segments of the original vector (called in MapSnap vector1), and
- Avoiding duplicating road points on the edges of the tiles,
- Avoiding corrupting road geometry at the edges of the tiles.

When tiles are created each road that goes from tile to tile gets new points on the edges of the adjacent tiles and duplicated (one point for the first tile and one point for the second tile). The algorithm starts from these edge road points, finds nearest turning points on the same road in both adjacent tiles and connects these turning points by a straight line. In this way, MapSnap preserved the topology and straight roads of vector 1 as well as removes duplicated road points on the edge of the adjacent tiles. The Tile Linking Algorithm (TLA) links the adjacent tiles transformed by MapSnap that have been approved by the user during the user inspection stage.

A user has two options to use MapSnap:

1) running MapSnap tile-by-tile with analysis and interactive adjustment of results after each tile and
2) running it automatically on all tiles and then adjusting results of all tiles.

In the first option, MapSnap links tiles and produced a vector 3 of linked tiles only after the user adjusted and approved each tile result going tile-by-tile. In the second option, MapSnap links automatically tiles produced only by ASA algorithm. In this way, MapSnap generates a linked vector 3 file. Then the user can inspect vector 3 if the user is not satisfied the user can analyze results of ASA relative to STLF for each tile, select the best of these results and/or correct them. Then MapSnap will create a file of selected ASA and STLF results and tiles corrected by the user. After that MapSnap links these tiles to a new vector 3. The first option is more interactive and can take more time from the user but can produce higher quality alignment eliminating tile overlaps or tile gaps. At this time, more complex tile linking is left to the manual work of the user. The next version of MapSnap will have a wizard that will guide a user in:

- finding significant overlaps and gaps between tiles,
- alerting a user that a manual adjustments of some specific tiles are needed, and
- guiding a user in this adjustment process (ordering and showing inconsistent tiles and links between tiles).

Dynamic Tiling. MapSnap supports assigning tile size of the image and vector data dynamically relative to the image size (in pixels) and resolution (in meters). Thus, the tile size is different for images with different size and resolution. MapSnap has two options for the user: (1) selecting tile size manually or (2) use the tile size that MapSnap computes automatically. We found that only experienced users do (1) successfully, therefore the most recent MapSnap implementation does this automatically. This decreases user's load for tuning MapSnap.

Example. Consider a raster file the coved the area of 3000 m x 3000 m with 1 m/pixel resolution. The largest tile size that MapSnap uses is 1000m ×1000m at this resolution. Thus, MapSnap produces 9 tiles of size 1000m × 1000m from this raster file. For a raster file the covers the area of 3200 m × 3000 m with 1 m/pixel resolution MapSnap produces 12 tiles of size 800m × 1000m from this raster. The tiling design allows MapSnap to work with images of any size within computer memory limitations.

5. TIME AND ACCURACY COMPARISON

The goal of this section is comparing time to conflate a misaligned vector file using MapSnap with time to digitize roads manually from the image using ArcMap editor. We conducted three experiments.

5.1. Experiment 1: MapSnap vs. manual digitizing roads

A geospatial scientist who is familiar with conflation issues conducted this experiment. He is a good representative of the targeted user community. As expected users, he does not know details of MapSnap algorithms and needs to learn the efficient ways of using MapSnap. An important factor was the use of remote parallel viewing of the user's computer window by one of the developers. This allowed an on-line instantaneous collaboration and greatly facilitated the testing process and setting up the MapSnap parameters.

First, the tester manually digitized the image, at 1:5000 scale. The extent of this image is shown in Figure 5 that covers area about 2.5×2.5km. The manual digitizing took *56 minutes*. The digitization process did not use node-snapping. Thus, the topology of the result is not complete. The result of this digitization was a bit less precise result than the original vector 1 because it was digitized at 1:5000.

The roads extracted by ACE algorithm have been given to the tester. The tester runs MapSnap on the same area as he used to digitize roads. MapSnap configuration was set by him. This mode of operation runs SLTF on the entire area and outputs an SLTF result (*global STLF*). This means that tiles where not used to make local transform for each tile. The entire area was processed in *4 minutes and 30 seconds*, of which setup and assessment time was *1 minute and 40 seconds.*

The STLF results in the NE and SE corners of the study area were not suitable. Therefore, the tester attempted to improve the results in these areas by editing vector 2 and rerunning STLF. Editing took *3:45 min*. The STLF results were better, but still not fixed completely so the editing was finished by hand editing. The problem segments editing time was *7:04 minutes* of a total time of *13:37 minutes*. These results are summarized in Table 1. The alignment using MapSnap *was 4 times faster* that digitization from scratch, 13 min 37 sec vs. 58 min 07 sec.

Table 1. User time in experiment 1

Time to	**MapSnap**	**Digitize**
Assess 1	1:40 min	N/A
Edit 1	3:45 min	N/A
Assess 2	1:08 min	N/A
Edit 2	7:04 min	56:07 min
Total	**13:37 min**	**56:07 min**

Table 2. User time in experiment 2

Description	time	Comment
Assess Tile 1	1 min	Inspection of ASA and STLF results
Assess Tiles 2-5,7-9	1 min each tile, total 7 min	Inspection of ASA or STLF results. Acceptable results without editing and rerun
Edit Tile 6	7 min	Both ASA and STLF produced inacceptable results. Editing and rerun was required to produce acceptable results
Assess Tile 6	2 min	
Total	**17 min**	

The digitization would take additional 20-60 minutes if it would be done at 1:1500 scale that the high-resolution image supports. In fact, the tester digitized the image at 1:5000. No attempts were made to attribute the digitized roads. This activity would add ~60-100 minutes for this area, while conflation can cut this time by transferring and editing attributes from vector 1 that MapSnap transforms.

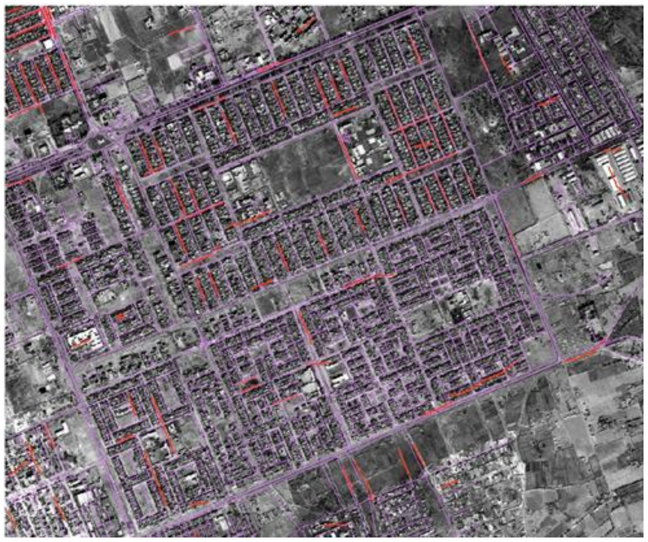


Figure 5. Extent of testing image (~ 3km × 3km) with automatically extracted roads (red)

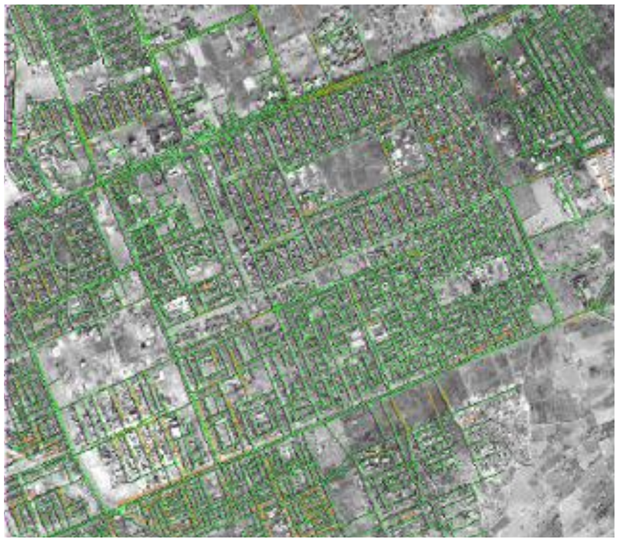


Figure 6. Global transform result-STLF (green)

5.2. Experiment 2: MapSnap with AFE and partial manual digitizing roads

The goal of this experiment is to evaluate total conflation time when MapSnap does not produce a good result in the first run. Specifically we consider a case when ASA algorithm has no enough data for confident conflation due to small tile size of the tile 300x200 in the initial MapSnap setting used. In this case, a user has two options: (1) adding missing roads manually (semi-manually) using ArcMap editor or another tool and rerun ASA, and (2) increase size of tiles.

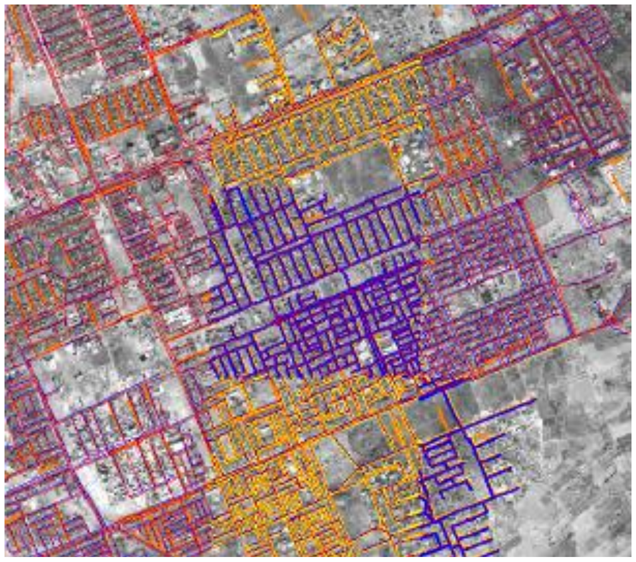


Figure 7. Tiling final result: 9 tiles

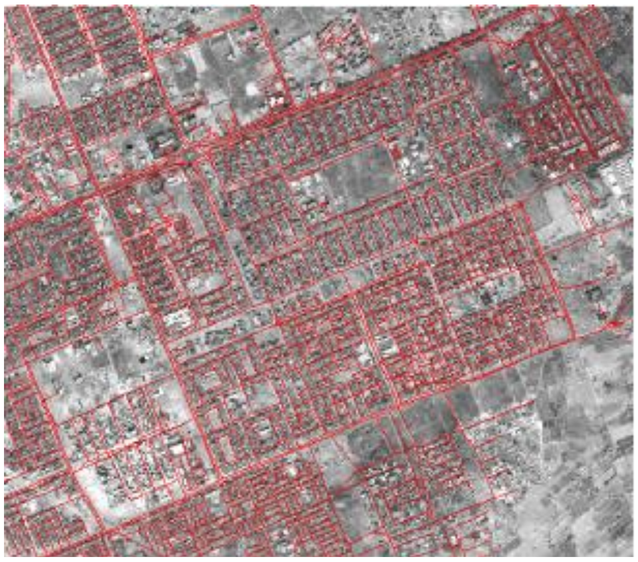


Figure 8. ASA tiles linked

The experiment 2 was conducted on data with increased tile size from 300x200 to 427x390 with total 9 tiles of this size and with few roads added manually by the user later in the course of the experiment. ASA produced a complete result from for all 9 tiles with *17 min* of the user time (10 min for inspection + 7 min for editing). Details of this experiment are in Table 2. As this table shows, only tile 6 required extensive manual work (7 min for editing), other tiles did not require editing of AFE results. Thus the total user time for manual editing is the same 7 min, which is about *two times less* than 13 min 37 sec observed in experiment 1. The smaller editing time is a result of combination of the use of *local transforms* in both STLF and ASA in contrast with the use of global STLF in experiment 1. Figure 7-14. show screenshots of representation results of experiment 2.

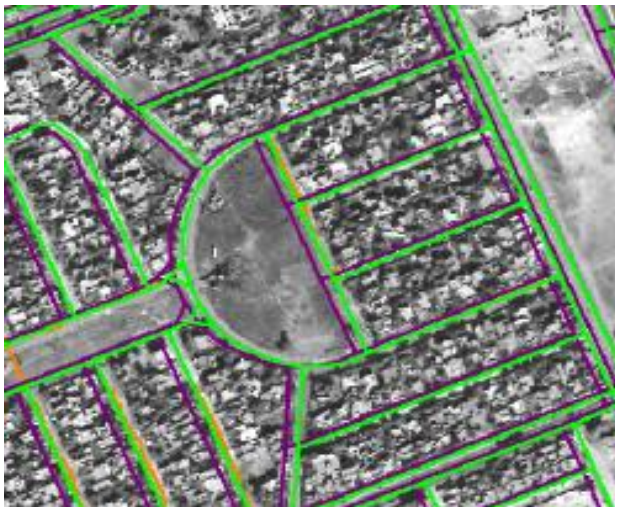


Figure 9.Zoomed Global transform (green) on tile 2

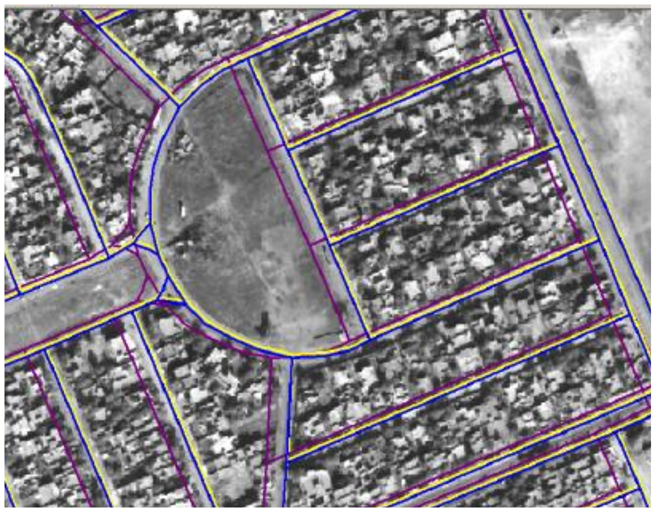


Figure 10. Zoomed ASA and STLF results on tile 2. Both are consistent with the global STLF transform and imagery. Magenta - original misaligned vector.

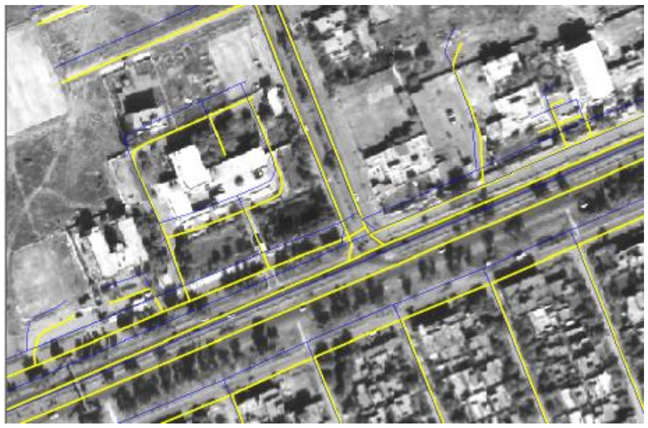


Figure 11. Tile 4: better aligned ASA result (yellow)

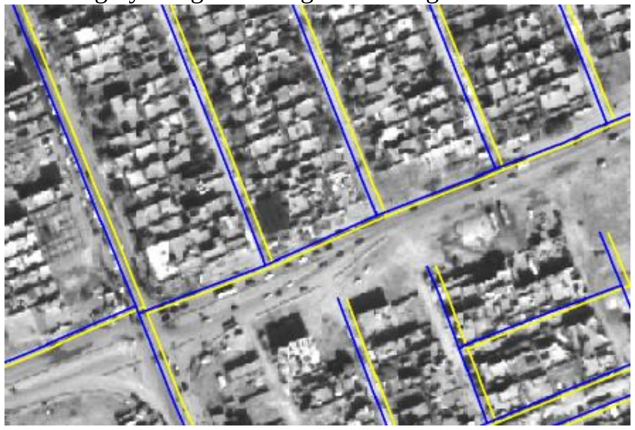


Figure 12 Tile 5: better aligned STLF result

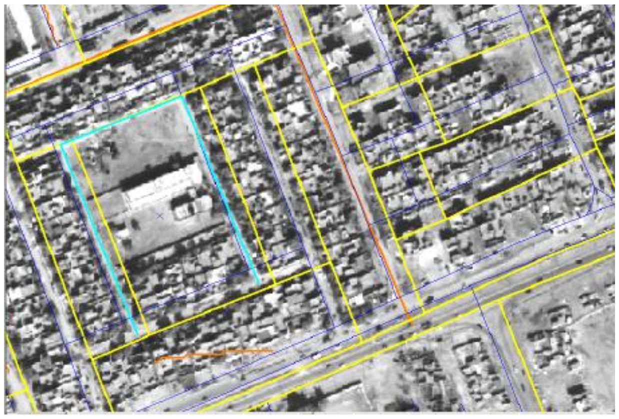


Figure 13. Tile 6: three lines added (blue) to vector 2 (extract) to improve ASA and STLF results

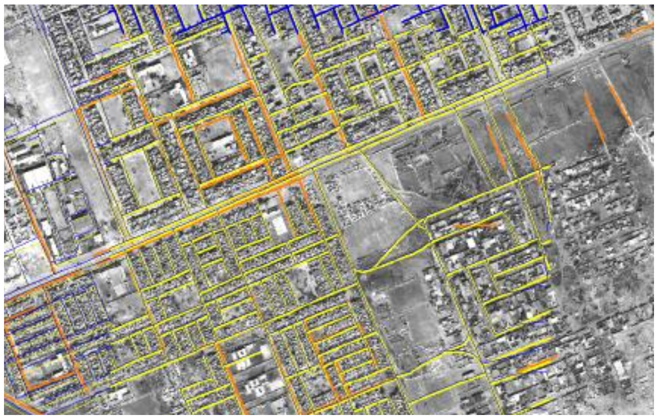


Figure 14. Fixed tile 6 after rerun on improved vector2 (extract)

5.3. Experiment 3: MapSnap with partial manual road digitizing and without AFE

In this experiment, AFE was not used by MapSnap. All extracted roads that MapSnap used were created manually by

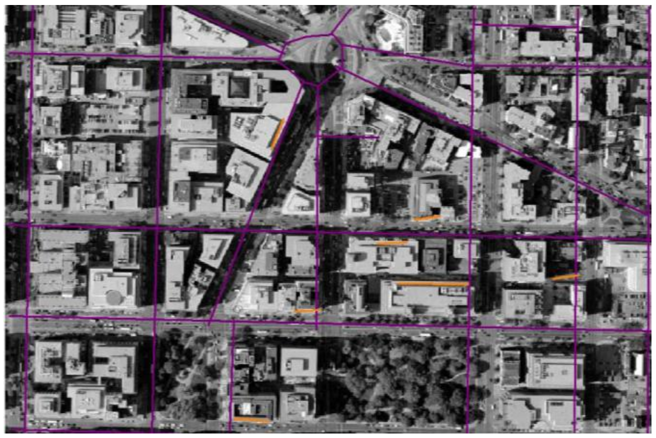


Figure 15. Vector map to be aligned (violet) and low quality automatic feature extract (ochre)

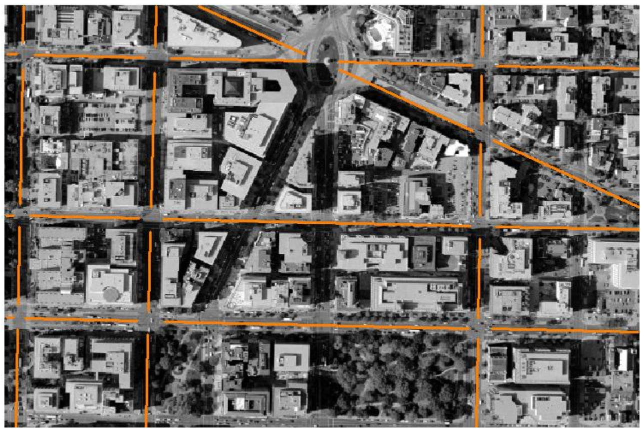


Figure 16. Extensive manual feature extract 1(orange)

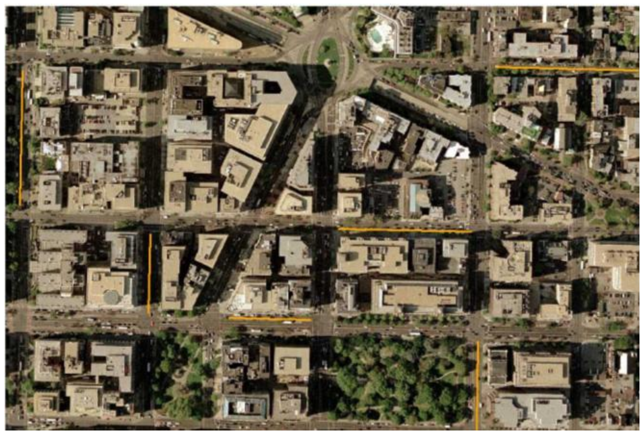


Figure 17. Minimal manual feature extract 2 (ochre)

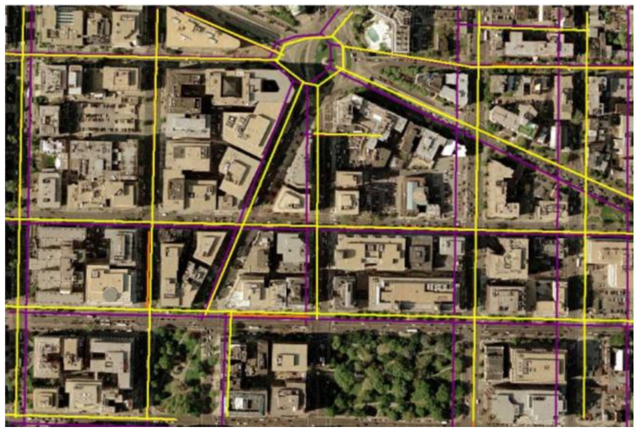


Figure 18. Corrected map using manual extract 2 (yellow)

the user using ArcGIS editor because of low quality of AFE (see Figure 15) where none of the extract was in the correct location. This experiment was conducted with ASA algorithm in two settings. In the *extensive manual road extraction* setting it took about 5 minutes to get 25 segments 9 (13 vertical segments, 9 horizontal segments and 3 diagonal segments). See Figure 16. In the *minimal manual road extraction* setting it took less than 1 minute to get 3 horizontal and three vertical lines (see Figure 17). Both produced the identical result with errors less than 3 meters (see Figure 18). Note that a simple shift is not sufficient for this data. These data require an affine transform where vertical lines of vector 1 (violet) on the right side should not move at all, but vertical lines on the left should move.

Other methods that rely on matching road intersection [7] will have difficulties to solve this task because road intersection are not extracted in Figures 15-17. This experiment had shows that the substitute of the AFE with minimal road extraction in Figure 17 is an efficient way to solve the alignment task that took *less than a minute of user's manual road extraction*.

6. CONCLUSION AND FUTURE WORK

While automated matching and fusing vector datasets has been a subject of research for years and strides are being made. However, much less has been done with matching or fusing vector and raster data. The MapSnap system has been developed for this purpose. Cutting time for alignment of existing vector road maps and imagery is a critical aspect of fusion of vector and raster geospatial data. The main components of the approach taken in MapSnap are: (1) data tiling, (2) combination of AFE with manual (semi-manual) feature extraction to get raw feature extracts, (3) use of two complimentary automatic conflation algorithms ASA and STLF applied to each tile, (4) linking transformed tiles to the integrated vector file, (5) multiple iteration of steps (1)-(4) to get a quality results. The experiments had show that MapSnap is able to cut time 4 times relative to manual road extraction from scratch. Experiments also had shown that substitution of fragile automatic feature extraction with *minimal manual feature extraction (MMFE)*/digitization have advantages for shortening time for correct aligning vector and raster data.

To enhance *minimal manual feature extraction* and to automate tuning MapSnapo parameters further research is needed. MMFE allows ensuring that extracts are correct and can be trusted. MMFE saves time to inspect and correct AFE results.

A specific direction to enhance MMFE and other aspects of MapSnap is developing *training tile approach*. In this approach a *Preprocessing step* where a user creates a "*ground truth*" vector file. In this tile, the user corrects/edits/shifts/transforms vector data and extracts features from the image manually or semi-manually using ArcGIS or other efficient editing tools. This will guide and optimize road extraction from other tiles, including enhancing MMFE, automating tuning MapSnap parameters, evaluating resolution differences between existing vector data and user needs. For instance, the comparison of the original vector road A with edited road A` can reveal that the`user digitized a road with 5 points that preserve road curvature, but vector 1 represents this road only with two points as a straight line. This tells that a simple shift most likely will not satisfy user's needs and a more sophisticated processing is needed.

Next, if almost every road in the training tile has its own unique transform that differs from transforms for other roads in the training tile then a very detailed and accurate feature extraction is required for these data. Knowing that AFE may not provide such accuracy, the time can be saved by doing this work differently. In contrast, if most of the roads in the training tile have similar transforms then the *generalization hypothesis* is confirmed and MMFE combined with less accurate AFE can be a good alignment strategy that will shorten the total processing time.

In summary, today MapSnap is most useful for the data that have either: (1) a single linear shift (STLF), (2) a set of different linear shifts, one shift per tile (STLF and ASA), and (3) a set of different affine transforms, one transform per tile (ASA). MapSnap is also useful for alignment of the vector data with imagery that do not satisfy exactly (1)-(3), but close enough to (1)-(3), i.e., alignment errors are within acceptable limits for the user's goals.

7. ACKNOWLEDGEMENTS

This work was funded by the U.S. Army Geospatial Center.

REFERENCES

[1] Doucette, P., Grodecki, J., Clelland, R., Hsu, A., Nolting, J., Malitz, S., Kavanagh, C., Barton, S., Tang, M. (2009). Evaluating Automated Road Extraction in Different Operational Modes. *Defense Security & Sensing*, Orlando, FL.

[2] Doucette, P., P. Agouris, and A. Stefanidis (2004). Automated Road Extraction from High Resolution Multispectral Imagery, Photogrammetric Engineering & Remote Sensing. 70(12), pp. 1405-1416

[3] Agouris, P., Stefanidis, A. & Gyftakis, S. (2001). Differential Snakes for Change Detection in Road Segments. *Photogrammetric Engineering and Remote Sensing* 67(12), 1391-1400.

[4] Kovalerchuk, B., Doucette, P., Brigantic, R., Seedahmed, G., Kovalerchuk, M., Graff, B. (2008). Automated Vector-to-Raster Image Registration. *SPIE Defense Security & Sensing*, Orlando, FL.

[5] Doucette, P., Kovalerchuk, B., Brigantic, R., Seedahmed, G., Graff, B. (2007). A Method for Vector-to-Image Registration. *Applied Imagery Pattern Recognition Workshop (AIPR 2007)*, IEEE Computer Society. (Washington, D.C.)

[6] Doucette P., Kovalerchuk, B., Kovalerchuk M., Brigantic R., An evaluation methodology for vector data updating, In SPIE Proceedings Vol. 7334, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XV, Shen S., Lewis P., Editors, 73341F

[7] Chen, C., Knoblock, C., Shahabi, C., Thakkar, C., Chiang Y., (2004). "Automatically and Accurately Conflating Orthoimagery and Street Maps", In: Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS'04)

[8] Mckeown, D., Bulwinkle, T., Cochran, S., Harvey, W., McGlone, C., Shufelt, J. (2000). Performance Evaluation for Automatic Feature Extraction. In: *International Archives of Photogrammetry and Remote Sensing*. ISPRS, Amsterdam, The Netherlands.

[9] Wiedemann, C., Heipke, C., Mayer, H. & Jamet, O. (1998). Empirical Evaluation of Automatically Extracted Road Axes. In: *Empirical Evaluation Methods in Computer Vision* (eds. Bowyer, K. & Phillips, P.), pp. 172-187, IEEE Computer Society Press.

[10] Kovalerchuk, B., Schwing, J., (Eds) Visual and Spatial Analysis, Springer, 2005.