
국토교통기술촉진연구사업 최종평가 검토자료

최종보고서 별첨자료

3차원 공간정보 제작을 위한 CPU병렬처리 시스템과
실시간으로 변화를 갱신할 수 있는 Local Dynamic Mapping 소프트웨어 개발

2022. 2. 24

- [1] S.Tareen, Z.Saleem (2018), “A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK”, International Conference on Computing, Mathematics and Engineering Technologies
- [2] C. Harris and M. j. Stephens (1988), “A combined cornerand edge detector”, In Alvey Vision Conference
- [3] J. Shi (1994), “Good Features to Track”, Conference on Computer Vision and Pattern Recognition
- [4] E. Rosten and T. Drummond (2006), “Machine learning for high-speed corner detection”, in ECCV
- [5] J. Matas, O. Chum, M. Urba(2002), “Robust wide baseline stereo from maximally stable extremal regions.” Proc. of British Marchine Vision Conference.
- [6] Lowe, D.G. (2004), “Distinctive image features from scale-invariant keypoints”, IJCV
- [7] D. Bouris, T. Tuytelaars, and L.V. Gool (2012), “SURF: Speeded Up Robust Features”, 9th European Conf. Computer Vision
- [8] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G.(2011), “ORB: an efficient alternative to SIFT or SURF,” in ICCV
- [9] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison (2012). Kaze features. In Computer Vision - ECCV
- [10] Pablo F Alcantarilla, Jesús Nuevo, and Adrien Bartoli (2011).“ Fast explicit diffusion for accelerated features in nonlinear scale spaces.” Trans. Pattern Anal. Machine Intell
- [11] S. Y. Shin, H. H. Han & S. H. Lee (2021). Improved YOLOv3 with duplex FPN for object detection based on deep learning. The International Journal of Electrical Engineering & Education
- [12] J. Long, E. Shelhamer, T. Darrell (2015), “Fully Convolutional Networks for Semantic Segmentation”, CVRP2015
- [13] Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla (2016), “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”, IEEE Transaction on PatternAnalysis and Machine Intelligence
- [14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015), “U-Net: Convolutional Networks for Biomedical Image Segmentation”, Medical Image Computing and Computer-Assisted Intervention (MICCAI)
- [15] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, Hartwig Adam (2018), “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”, In ECCV
- [16] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, Jiaya Jia(2017), “Pyramid Scene Parsing Network”, In CVPR
- [17] Changwoo Song, Wiratama Wahyu, Jihun Jung, Seongjae Hong, Daehee Kim (2020),“Urban Change Detection for High-resolution Satellite Images Using U-Net Based on SPADE”, Korean Journal of Remote Sensing (Korean J. Remote Sens.)

- [18] Samitha Daranagama, Apichon Witayangkurn (2021), “Automatic Building Detection with Polygonizing and Attribute Extraction from High-Resolution Images”, In ISPRS
- [19] Kang, S. M. & Lee, J. J (2019). “Depth Map Extraction from the Single Image Using Pix2Pix Model”, JOURNAL OF KOREA MULTIMEDIA SOCIETY
- [20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros (2017), “Image-to-Image Translation with Conditional Adversarial Networks”, In CVPR
- [21] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio(2014), “Generative Adversarial Networks”, Machine Learning
- [22] Tero Karran, Timo Aila, Samuli Laine, Jaakko Lehtinen (2017), “Progressive Growing of GANs for Improved Qualitym Stability, and Variation”, In ICLR
- [23] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, Bryan Catanzaro (2017), “High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs”, In CVPR
- [24] 조영주, 배강민 박종열 (2020), “GAN 적대적 생성 신경망과 이미지 생성 및 변환 기술 동향”, 전자통신동향분석
- [25] Sahu, M. & Ohri, A. (2019), “VECTOR MAP GENERATION FROM AERIAL IMAGERY USING DEEP LEARNING”, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences
- [26] Parathap, G. Afanasyev, I. (2018), “Deep learning approach for building detection in satellite multispectral imagery”, International Conference on Intelligent Systems
- [27] Chen Qi et al. (2018), “Aerial Imagery for Roof Segmentation: A Large-Scale Dataset towards Automatic Mapping of Buildings”
- [28] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, Pierre Alliez (2017), “Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark”, International Geoscience and Remote Sensing Symposium (IGARSS)
- [29] Ugur Demir, Gozdel Unal (2018), “Patch-Based Image Inpainting with Generative Adversarial Networks”, In Access
- [30] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, Stephen Paul Smolley(2016), “Least Squares Generative Adversarial Networks”, In ICCV
- [31] 박진혁, 장봉주, 권오준, 정재진, 이석환, 권기룡 (2015), “Douglas Peucker 근사화 알고리즘과 빈 분류 기반 벡터 맵 데이터 압축”, 멀티미디어학회논문지
- [32] DronesSurvey& 3D Mapping Solutions 교육 및 서비스 Droneuavman작성 NaverBlog 발췌
- [33] 현대자동차-기아(2019), “자동차 회사, 하늘길을 넘보다. 미래 도시 교통난을 해소할 UAM”, HMG Journal

별첨2

총 연구개발기간 성과점검기준표 및 증빙자료

별첨 2-1. 총 연구개발기간 성과기준점검표

[총괄] 성과점검기준표

(과제명 : 3차원 공간정보 제작을 위한 CPU 병렬처리 시스템과 실시간으로 변화를 갱신할 수 있는 Local Dynamic Mapping 소프트웨어 개발)

구분	핵심연구성과(level1)		단위성과물 (level2)		최종 성과점검기준										
					성과검증 기준						전문기관 검토의견 (인정/불인정/일부미흡)	해당기관	가중치 (세부별)	달성평점 (평점 :1~5)	점수 (100점 만점)
					질적성과지표	목표수준		측정방법	증빙						
						목표치	달성치								
주관 (건국대)	A	CPU 병렬처리 시스템	A-1	H/W 설계 및 보고서 제출	①	최적 CPU 수 결정	1	1	보고서 실험자료	증빙자료집 P.1~3		건국대학교 신학협력단	3.3		0.00
			A-3	시제품	②	상품화 가능제품	1	1	심사	증빙자료집 P.11~14		건국대학교 신학협력단	3.3		0.00
	B	Local Dynamic Mapping S/W개발	B-1	일반 Pair 사진으로 tie point 추출	①	10 point / Pair	1	1	심사	증빙자료집 P.23~25		건국대학교 신학협력단	1.2		0.00
			B-2	항공사진에서 tie point 추출	②	10 point / Pair	1	1	심사	증빙자료집 P.23~25		건국대학교 신학협력단	3.8		0.00
	C	지형지물(건물) 자동인지 AI 알고리즘 개발	C-1	지형지물 인지 가능 연구 보고서 제출	①	보고서 심사	1	1	보고서 심사	증빙자료집 P.7~8		건국대학교 신학협력단	1.0		0.00
			C-2	지형지물(건물) 자동인지 알고리즘 개발	②	60% 인지율	60%	95%	심사	증빙자료집 P.26~32		건국대학교 신학협력단	3.7		0.00
공동 (㈜엠지아이에스)	A	CPU 병렬처리 시스템	A-2	자동화 S/W 설계 및 제작	①	처리시간 1/10	1/10	1/10	심사	증빙자료집 P.9		㈜엠지아이에스	1.2		0.00
	C	지형지물(건물) 자동인지 AI 알고리즘 개발	C-3	인공지능 알고리즘 이용한 S/W 제작	②	60% 인지율	60%	95%	심사	증빙자료집 P.10		㈜엠지아이에스	2.5		0.00
												20.0	-	0	
양적 달성 정도	목표						목표 건수	달성 건수	증빙		인정 건수	달성율 (%)	가중치(%)	점수 (100점 만점)	
	사업연계지표	1	특허 출원 건수				2	2	증빙자료집 P.15~22			0	27.5	0	
		2	국내·외 학술지 게재논문 건수 (SCI)				2	2	증빙자료집 P.1~8			0	32.5	0	
		3	시제품 제작				1	1	증빙자료집 P.11~14			0	12.5	0	
		4	소프트웨어 등록 건수				2	2	증빙자료집 P.9~10			0	27.5	0	
													100.0	0	

[1차년도] 성과점검기준표

(과제명 : 3차원 공간정보 제작을 위한 CPU 병렬처리 시스템과 실시간으로 변화를 갱신할 수 있는 Local Dynamic Mapping 소프트웨어 개발)

구분	핵심연구성과(level1)		단위성과물 (level2)		최종 성과점검기준										
					성과검증 기준						전문기관 검토의견 (인정/불인정/일부미흡)	해당기관	가중치 (세부별)	달성평점 (평점 :1~5)	점수 (100점 만점)
					질적성과지표	목표수준		측정방법	증빙						
						목표치	달성치								
주관 (건국대)	A	CPU 병렬처리 시스템	A-1	H/W 설계 및 보고서 제출	①	최적 CPU 수 결정	1	1	보고서 실험자료	증빙자료집 P.1~3		건국대학교 신학협력단	6.6		
			A-3	시제품	②	상품화 가능제품	1	1	심사	증빙자료집 P.11~14		건국대학교 신학협력단	6.6		
	B	Local Dynamic Mapping S/W개발	B-1	일반 Pair 사진으로 tie point 추출	①	10 point / Pair	1	1	심사	증빙자료집 P.23~25		건국대학교 신학협력단	2.4		
	C	지형지물(건물) 자동인지 AI 알고리즘 개발	C-1	지형지물 인지 가능 연구 보고서 제출	①	보고서 심사	1	1	보고서 심사	증빙자료집 P.7~8		건국대학교 신학협력단	2.0		
공동 (㈜엠지아이에스)	A	CPU 병렬처리 시스템	A-2	자동화 S/W 설계 및 제작	①	처리시간 1/10	1/10	1/10	심사	증빙자료집 P.9		㈜엠지아이에스	2.4		
	C	지형지물(건물) 자동인지 AI 알고리즘 개발													
													20.0	-	0
양적 달성 정도	목 표						목표 건수	달성 건수	증빙		인정 건수	달성율 (%)	가중치(%)	점수 (100점 만점)	
	사업연계지표	1	특허 출원 건수				1	1	증빙자료집 P.15~22			0	25	0	
		2	국내·외 학술지 게재논문 건수 (SCI)				1	2	증빙자료집 P.1~6			0	35	0	
		3	시제품 제작				1	1	증빙자료집 P.11~14			0	25	0	
		4	소프트웨어 등록 건수				1	1	증빙자료집 P.9			0	15	0	
													100.0	0	

[2차년도] 성과점검기준표

(과제명 : 3차원 공간정보 제작을 위한 CPU 병렬처리 시스템과 실시간으로 변화를 갱신할 수 있는 Local Dynamic Mapping 소프트웨어 개발)

구분	핵심연구성과(level1)		단위성과물 (level2)		최종 성과점검기준										
					성과검증 기준						전문기관 검토의견 (인정/불인정/일부미흡)	해당기관	가중치 (세부별)	달성평점 (평점 :1~5)	점수 (100점 만점)
					질적성과지표		목표수준		측정방법	증빙					
목표치	달성치														
주관 (건국대)															
	B	Local Dynamic Mapping S/W개발													
			B-2	항공사진에서 tie point 추출	②	10 point / Pair	1	1	심사	증빙자료집 P.23~25	건국대학교 산학협력단	7.5	5	37.50	
	C	지형지물(건물) 자동인지 AI 알고리즘 개발													
C-2			지형지물(건물) 자동인지 알고리즘 개발	②	60% 인지율	60%	95%	심사	증빙자료집 P.26~32	건국대학교 산학협력단	7.5	5	37.50		
공동 (㈜염지아이에스)															
	C	지형지물(건물) 자동인지 AI 알고리즘 개발	C-3	인공지능 알고리즘 이용한 S/W 제작	②	60% 인지율	60%	95%	심사	증빙자료집 P.10	㈜염지아이에스	5.0	5	25.00	
												20.0	-	100	
양적 달성 정도	목 표					목표 건수	달성 건수	증빙		인정 건수	달성율 (%)	가중치(%)	점수 (100점 만점)		
	사업연계지표	1	특허 출원 건수			1	출원 진행 중	-			0	30	0		
		2	국내·외 학술지 게재논문 건수 (SCI)			1	게재 예정	증빙자료집 P.7~8			0	30	0		
		3	소프트웨어 등록 건수			1	1	증빙자료집 P.10			0	40	0		
												100.0	0		

별첨 2-1. 증빙자료집

번호	관련증빙 명
1	[논문] 3차원 공간정보 제작을 위한 CPU 병렬처리 시스템에 관한 연구
3차원 공간정보 제작을 위한 CPU 병렬처리 시스템에 관한 연구 Study of parallel processing system to build 3D Geospatial information 서정현*, 김상호**, 윤준성*** Jeongheon Seo, Sangho Kim, Junseong Yun 건국대학교 스마트운행체공학과 초빙교수(a21689@konkuk.ac.kr) 건국대학교 스마트운행체공학과 교수(kimsh85@konkuk.ac.kr) 건국대학교 스마트운행체공학과 석사과정(springboat@konkuk.ac.kr) 요 약 본 연구에서는 드론측량을 위한 Photogrammetry s/w중에서 network을 이용하여 병렬처리를 수행할 수 있는 Agisoft사의 Metashape를 기반으로 자료처리 시간을 단축하기 위한 실험을 수행하였고 그 실험결과로 가격대비 성능비가 최적이 되는 Node의 수를 결정하였다. 특히 본 실험을 통하여 client-node 방식의 병렬처리 방식만으로 자료처리 시간을 1/10로 줄일 수 있다는 것을 알 수 있었다. 1. 서론 드론을 이용한 사진측량은 값싼 가격과 운용 및 자료처리의 용이성 등으로 인하여 범용화 되고 있는 추세이다. 대다수 광범위하게 이용되고는 있으나 아직 드론 측량이 산업화 되어서 한 축을 담당하고 있지 못한 이유는 아래의 3가지 측면이 있다. 첫째 드론의 비행시간과 비행거리의 제한으로 인하여 공간정보 제작 품셈의 기본단위인 1km2조차도 1 sortie에 Data 획득을 끝내지 못하며 둘째 사진 측량의 기본 값인 렌즈의 중심좌표와 사진의 기울기 값을 드론의 Flight Controller로부터 제공받고 있으나 측량 결과의 정밀도와 정확도를 보장하기에는 부족하다. 마지막으로 이를 극복하기 위하여 H/W적으로 그리고 S/W적으로 다양한 보완책이 제시되고 있으나 시장에서 쉽게 획득 할 수 있는 방안은 아직 찾기 힘들다. 본 연구의 사전 연구를 통하여 산출물의 정확도를 보장할 수 있는 고가의 INS와 정밀도가 보장되는 카메라를 결합하여 드론측량용 카메라를 제작하여 공인 받았으며, S/W적으로 사진의 밀도를 높여서 한 점의 point 값을 측정하는데 5! 수만큼의 pair(120pair)를 중복 계산하여 정확도 2cm를 달성하였다. 본 연구에서는 위의 카메라로 촬영된 6700매의 (90%의 중복도) 사진을 고속으로 처리하기 위하여 CPU 병	
- 166 -	

열처리 방식을 실험하였다.

2. 본론

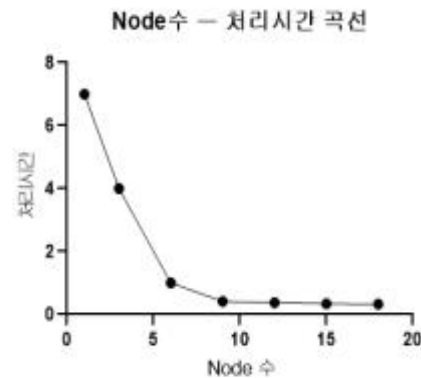
항공상각측량은 단순반복계산이기는 하나 그 계산횟수가 매우 커서 자료처리에 많은 시간이 소요 된다. 특히 카메라의 화소수가 증가하고 정확도를 향상시키기 위한 사진의 밀도가 증가하면서 통상 1 Batch로 처리하는 사진의 매수가 적게는 수천 매에서 많게는 수만 매에 이르며 물리적 용량 역시 수십Gbyte에 이른다. 특히 측량용선의 기본단위인 1km^2 를 측량하기 위해서는 고도 60m-100m 비행할 경우 사진의 양은 7천매 ~ 만매에 이른다.

이런 대량의 사진을 신속하게 처리하기 위해서 가장 좋은 방법은 병렬처리를 하는 것이다. 본 연구에서는 client-node의 조합으로 CPU 병렬처리하는 방식을 실험하였다.

자료처리 전체 과정에서 server에 의지한 client는 같은 반복 계산을 여러 대의 node에게 나누어 주어 처리시키고 그들 종합하여 결과 값을 내도록 하는 분산병렬처리 방식을 이용하였다. 실험의 목적은 무한히 많은 node를 이용하는 것은 비현실적이기 때문에 적당한 node의 수를 찾아내기 위함이었다. 단 내부망의 속도는 기가 인터넷을 기준으로 하였고 다양한 속도의 Network은 실험하지 않았다. 처리한 시간을 기준으로 node의 수는 3의 배수로 설정하였고 3대로부터 최대 18대까지 실험하였다. 실험한 결과 값은 아래의 표와 같다. 실험결과 곡선의 변곡점인 8개의 node수가 가격대비 성능의 최적 수임을 알 수 있었다. 또한 처리시간 역시 1/10 이상으로 단축시킬 수 있었다.

표 1 가격대비 성능비 계산표

node 수	최종 산출물 산출시간	가격대비 성능변화
18	8시간	0.012
15	8.5시간	0.03
12	9시간	0.041
9	10시간	0.58
6	1일	3
3	4일	4.5
1(stand alone)	7일	기준



자료처리조건

1. CPU: i5, RAM: 16Gbyte, No GPU
2. 대상면적: 0.65km^2 , 사진매수: 6,700매
3. 처리option: 정확도 2cm

실험에 사용된 컴퓨터의 H/W는 CPU : i5, RAM : 16Gbyte이고, CPU병렬처리의 효과만을 알기 위하여 GPU는 설치하지 않았다. 총 20대의 컴퓨터를 기존의 학교 기가인터넷 망에 연결하였고, 1대의 server, 1대의 client와 18대의 node로 구성하였다. Network의 속도 역시 영향이 클 것으로 판단되었으나 그 영향이 비례적일 것으로 사료되어 다양한 속도의 Network를 이용한 실험은 하지 않았다. i5 CPU와 i7

CPU, 그리고 GPU 2장과 GPU6장에 대한 실험도 수행하였으나 별 성과 없었으며 이유는 현재 가용한 대부분의 사진측량 S/W들이 GPU coding을 하고 있다고는 하나 실제 실험결과에 의하면 아주 적은 부분에서 극히 일부만 GPU를 사용하고 있는 것으로 사료 되었다. 이는 앞으로 전면적인 GPU coding이 된 S/W를 개발한다면 매우 경쟁력이 있을 것을 시사한다.

획득된 사진은 총 6,700매로서 비행고도 60m, 렌즈 초점거리 16mm이며 촬영 중복도는 90% 이상으로 사진이 획득되었다. 자료처리 option은 정확도 2cm를 보장할 수 있도록 최대한 정밀하고 정확도를 높일 수 있는 option이 선택되어 자료처리 되었다.

option 중 중요한 요소들은 다음과 같다.

중요한 option

1. Alignment : highest accuracy & Alignment Optimization(Re-Alignment)
2. Point Cloud : ultra high quality
3. Mosaic : 정밀도 2cm

3. 결론

항공사진측량의 자료처리 과정은 단순반복계산의 연속으로서, 병렬처리를 한다면 처리속도를 기하급수적으로 단축할 수 있으나 병렬의 수는 최적의 수가 존재하며 무한히 증가하지 않는다는 것을 알 수 있었다.

본 연구에서는 현재의 조건하에서 8개의 Node수가 가격대비 최적의 수인 것으로 판단되었으며 고속의 CPU를 사용하고, 더 많은 RAM을 장착하고, 더 빠른 Network를 사용하는 등 더 성능이 좋은 H/W를 사용하면 처리시간은 더욱 단축될 것이며 현장에서 요구하는 조건인 1km²의 지역에 대한 공간정보 생산 시간을 4시간이내로 단축하는 것이 가능할 것으로 판단된다.

감사의 글

본 연구는 국토교통부/국토교통과학기술진흥원의 지원으로 수행되었음(과제번호 20CTAP-C158040-01)

참고문헌

1. Agisoft, 2020, Agisoft Metashape Network Version, <http://www.agisoft.com/>

번호	관련증빙 명
2	[논문] 식생지수 판단에 최적화되어있는 다중분광 센서를 이용한 하천의 녹조탐지 가능성 연구
식생지수 판단에 최적화되어있는 다중분광 센서를 이용한 하천의 녹조탐지 가능성 연구 A Study on possibility of detecting algae in river using multi-spectral sensor optimized for determination of vegetation index. 서정현*, 김상호**, 윤준성*** Jeongheon Seo, Sangho Kim, Junseong Yun 건국대학교 스마트운행체공학과 초빙교수(a21689@konkuk.ac.kr) 건국대학교 스마트운행체공학과 교수(kimsh85@konkuk.ac.kr) 건국대학교 스마트운행체공학과 석사과정(springboat@konkuk.ac.kr) 요 약 다중분광 센서를 활용한 원격탐사는 환경관리, 공간모니터링, 탐지 등 다양한 분야에서 활용되며 드론용 분광 센서 개발 이후로 특히 정밀농업 분야에 활발하게 적용되고 있다. 다중분광 영상에 정밀농업에 사용되는 식생지수 적용하면 해당 지역의 식생 건강도 및 엽록소 활성도를 파악할 수 있다. 하천 및 호수 지역에 문제가 되는 녹조현상을 발생시키는 조류 역시 엽록소를 가지기 때문에 다중분광 센서를 활용한 영상에 식생지수를 적용한다면 해당 지역의 녹조가 밀집해 있는 부분을 찾을 수 있다. 본 연구에서는 다중분광 센서를 활용하여 만든 정사영상에 ARVI, NDVI, GARI, VARI 등 여러 식생지수를 적용하여 녹조를 탐지하였으며 식생지수 중 ARVI, VARI 적용했을 때 가장 직관적으로 녹조가 밀집된 지역을 찾을 수 있었다. 1. 서론 여름철 발생하는 녹조현상은 햇빛 차단으로 인한 수생태계 파괴 및 각종 냄새 물질과 독소 생성으로 불쾌감을 주는 등 다양한 문제를 일으킨다. 이러한 녹조 현상의 원인인 남조류는 광합성 색소인 엽록소 가진 녹색 조류를 말하며 부영양화된 호수, 유속이 느린 하천 등에서 크게 늘어나 하천 및 호수 등이 녹색이 되는 현상을 발생시킨다. 넓은 면적을 관측하며 자동으로 분석 가능한 분광영상은 당수조사 분야에서 기존 드론 촬영 영상이 빛의 정반사에 의해 의미있는 데이터를 생성하지 못하는 반면 다중분광 센서를 통해 촬영된 영상의 경우 다양한 분광반사 특성을 파악할 수 있기 때문에 수체, 식생, 토양 등의 분석이 가능하다. 또한 정밀농업 분야에서 다중분광 센서 영상에 식생지수를 적용하여 녹색식물의 상대적 분포량, 엽록소 함량 등의 지표로 사용되어 해당 지역 및 식물의 건강상태를 파악할 수도 있다. 본 연구에서는 드론에 다중분광 센서를 장착하여 대상 지역을 촬영하고, 정사영상(Orthomosaic map)을 제작하여 후처리로 식생지수를 적용해 대상 지역의 엽록체 활성도를 파악하여 녹조 밀	
- 95 -	

집 지역을 탐지하였다.

2. 본론

2.1 대상 지역 선정 및 미션 플래닝

건국대학교 서울캠퍼스 내 일감호는 저수 용량보다 유입 수량이 적어 물이 장기간 체류되며, 복합적 요인들에 의하여 부영양화되어있어 담수 조류가 과다 증식하기 좋은 환경이다. 또한, 해마다 녹조현상이 발생하며 면적이 55,661㎡에 달하여 녹조가 밀집된 지역을 구분하기 용이하여 대상 지역으로 선정하였다. 본 연구에서는 DJI사의 Phantom 4 드론에 5개의 파장을 가진 Micasense사의 Rededge-M 다중분광 센서를 탑재하여 운영하였다. 다중분광 센서와 DJI - Phantom 4의 규격에 맞추어 설계한 탑재용 부품을 3D프린터를 사용해 제작하였으며 Rededge-M의 GPS와 DLS Port는 드론의 비행 방향에 맞추어 드론 상단부에 부착하였다. 촬영 간격의 경우 WiFi 연결을 통하여 1초 간격으로 각 파장당 1장씩 총 5장의 사진을 찍도록 설정하였다.



그림 1 사용 장비(다중분광센서 마운팅)

미션 플래닝은 녹조의 이동 가능성을 고려하여 수립하였다. 녹조는 유동체로 시간에 따라 위치와 양이 변하므로 정사영상의 제작을 위한 최소의 중복도를 주어 빠른 시간 내에 일정 정확도를 가지는 정사영상 제작을 목적으로 비행하여 단시간 내에 결과물 파악에 주력하였다. 대상 지역인 물은

반사률이 매우 크기 때문에 비행고도 100m로 비행하였다.

드론 비행을 통해 1280x960 해상도의 Blue, Green, Red, NIR, Rededge 각 대역별 647매의 사진을 획득하였다. 획득한 영상들을 활용하여 식생지수를 직관적으로 잘 나타낼 수 있는 디지털 사진측량 프로그램인 Agisoft사의 Metashape를 사용하여 정사영상을 제작하였다.



그림 2 제작된 정사영상

2.2 식생지수(Vegetation Index) 적용

다중분광 센서를 사용하여 촬영한 분광 영상으로 제작된 정사영상에 ARVI, EVI, GARI, GNDVI, IPVI, MSR, NDRE, NDVI, OSAVI, SAVI, TDVI, VARI 총 12종류의 식생지수를 적용하였다. 위 식생지수 중 대기 요인에 저항성이 있어 에어로졸 함량이 높은 지역에 적용될 수 있는 ARVI와 VARI를 적용한 결과 정사영상에서 녹조 밀집지역인 A, B 지역이 가장 가시적으로 잘 나타났다으며 각 적용 수식은 다음과 같다.

$$ARVI = \frac{NIR - [Red - \gamma(Blue - Red)]}{NIR + [Red - \gamma(Blue - Red)]}$$



그림 3 Atmospherically Resistant Vegetation Index(ARVI)

$$VARI = \frac{Green - Red}{Green + Red - Blue}$$



그림 4 Visible Atmospherically Resistant Index(VARI)

실제 A, B 해당 지역을 관측한 결과 녹조가 밀집되어있는 것이 가시적으로 드러나 있어 위 ARVI, VARI 식생지수 적용을 통해 나타난 부분이 녹조가 밀집된 지역임을 확인할 수 있었다.

3. 결론

본 연구에서는 식생 상태를 판단하기에 최적화되어있는 다중분광 센서를 활용한

녹조탐지방안을 제시하였으며 대상 지역인 밀강호 지역을 드론 촬영을 통한 정사영상 제작과 식생지수를 적용을 통하여 녹조 집중지역을 탐지하였다. 이때 사용된 식생지수 중 대기 요인에 저항성이 있는 ARVI와 VARI에서 녹조 집중지역이 가장 잘 나타났다. 해당 결과는 정밀농업에서 녹색식물의 분포 또는 식생의 엽록소 함량 검출 등에 사용되던 식생지수를 활용해 간편하게 녹조 집중지역 탐사를 가능하게 할 것이지만 녹조의 이동 가능성이 있는 하천에 적용하여 밀집 지역을 찾을 시 정사영상 제작 시간 단축에 관한 문제를 먼저 해결해야 할 것이다.

감사의 글

본 연구는 국토교통부/국토교통과학기술진흥원의 지원으로 수행되었음(과제번호 20CTAP-C158040-01)

참고문헌

1. 환경부, 녹조(綠), 녹조현상은 무엇인가?
2. Agisoft 2020, Agisoft metashape (www.agisoft.com)
3. Kaufman, Y., and D. Tanre.1992 "Atmospherically Resistant Vegetation Index (ARVI) for EOS-MODIS, IEEE Transaction on Geoscience and Remote Sensing 230, no. 2(1992): 261-270.
4. Gitelson, A., et al. 2002. "Vegetation and Soil Lines in Visible Spectral Space: A Concept and Technique for remote Estimation of Vegetation Fraction," International Journal of Remote Sensing 23
5. 이강원, 손호웅, 김덕인 2016, 드론(무인기) 원격탐사 사진측량
6. Rededge-M User Manual (www.micasense.com)

번호	관련증빙 명
3	[등록예정] 드론정사영상을 이용하여 인공지능 신경모델로 벡터지도를 자동생성하는 기법
드론정사영상을 이용하여 인공지능 신경모델로 벡터지도를 자동생성하는 기법 Vector map auto generating technique using UAV photogrammetry and artificial neural network 윤준성* · 서정현** · 김상호*** Junseong Yun · Jeongheon Seo · Sangho Kim* 건국대학교 스마트은행채공학과 석사과정 (springboat@konkuk.ac.kr) 건국대학교 스마트은행채공학과 초빙교수 (a21689@konkuk.ac.kr) 건국대학교 스마트은행채공학과 교수 (kimsh85@konkuk.ac.kr) 요 약 최근 위성영상, 항공사진 등 영상지도로부터 인공지능을 활용하여 벡터형태의 공간정보를 추출하는 연구가 활발히 진행되고 있다. 그러나 드론을 활용한 무인항공사진측량으로 제작된 수cm의 지상표본거리를 가지는 정사영상은 위성영상, 항공사진 대비 지형지물이 차지하는 면적이 커 벡터형태의 공간정보 추출에 어려움이 있다. 본 연구에서는 적대적 생성 신경망 구조의 이미지 변환 인공지능 모델을 활용하여 드론정사영상으로부터 벡터지도를 자동생성하는 기법을 제시하고 도심지의 주요 지형지물 중 하나인 건물의 벡터를 자동 생성하였다. 1. 서론 최근 인공지능을 활용하여 영상정보로부터 건물, 도로, 산림지역 및 농경지 등 다양한 공간정보를 추출하는 연구가 이루어지고 있다. 이러한 인공지능 기반의 영상분할을 활용한 영상지도로부터 지형지물을 검출하는 연구들은 대부분 위성영상 또는 항공사진을 대상으로 하고 있다. 드론을 활용한 공간정보 생성방식은 30~150m의 근접촬영을 통하여 자료를 획득하기 때문에 앞의 두 방식 대비 정확한 정사영상 제작이 가능한 이점이 있으나 기존의 위성영상 등에 적용되는 인공지능을 활용한 지형지물 검출 방법 적용 시 지상표본거리가 상대적으로 작아 지형지물들이 차지하는 면적이 큰 드론정사영상으로부터 대상물의 추출이 어렵다. 본 연구에서는 수cm의 지상표본거리를 가지는 고해상도의 드론정사영상으로부터 적대적 생성신경망 구조의 인공 신경 모델을 활용하여 고해상도 이미지변환으로 지형지물을 검출하고 검출된 지형지물을 활용하여 벡터지도를 자동으로 생성하는 기법을 연구하였다.	
- 1 -	

2. 본론

본 연구에서는 지상표본거리 5cm/pix의 드론정사영상에서 건물을 대상으로 벡터 지도를 자동 생성하였다. 벡터 지도 생성을 위하여 제안하는 절차는 그림 1과 같다.

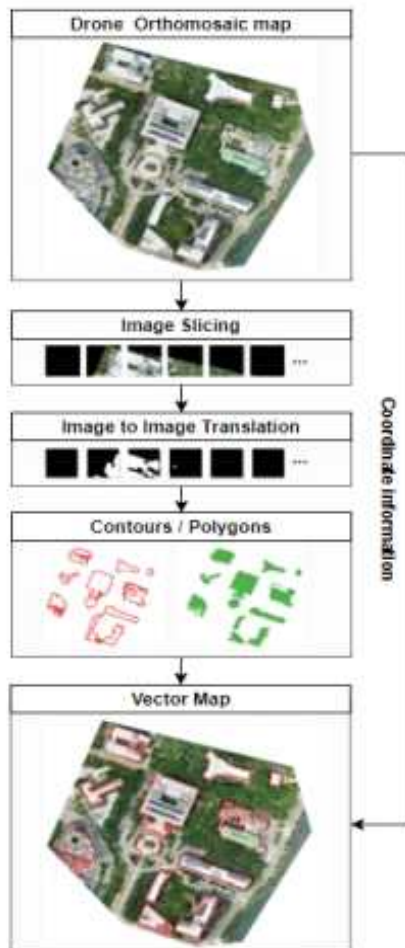


그림 1 Workflow

드론을 활용한 무인항공측량으로 생성된 정사영상을 이미지 변환 A.I모델의 입력 크기로 분할하고 각 이미지에 대하여 건물 영역만 분할된 이미지로 변환한다. 이미지 변

환 모델의 입력 크기는 1,024x1,024로 건물의 전반적인 형태가 이미지 내에 나타날 수 있는 크기로 선정하였다. 이미지 변환을 위하여 생성자와 판별자 두 개의 신경망이 경쟁하며 학습되는 적대적 생성 신경망(GAN) 구조의 네트워크를 구축하고 고해상도 이미지 변환 시 요구되는 깊은 구조의 네트워크에 발생하는 기울기 소실 문제에 안정적인 Multi-scale gradient를 사용하는 Unet 형태의 모델을 활용하였다[1].

이후 각 변환된 이미지들을 이어 붙여 정사영상과 동일한 크기의 변환 이미지를 생성하였다. 생성된 건물 분할 이미지로부터 건물의 외곽선을 추출하고, 이후 건물 외곽을 다각형(Polygon)형태로 추출하여 벡터 데이터를 생성한뒤 드론정사영상의 절대좌표 및 좌표계를 매핑하여 벡터지도를 생성하였다.

3. 결론

본 연구에서는 인공 신경망 모델을 활용하여 드론정사영상으로부터 벡터지도를 자동으로 생성하는 방법을 제안하였다. 추가적인 지형지물에 대하여 제시한 방법을 적용하여 건물만이 아닌 다양한 지형지물에 대한 벡터지도를 생성할 수 있을 것으로 기대된다.

감사의 글

본 연구는 국토교통부/국토교통과학기술진흥원의 지원으로 수행되었음(과제번호 : 21CTAP-C158041-02)

참고문헌

- [1] K. Laxman, et al. "Efficient High-Resolution Image-to-Image Translation using Multi-Scale Gradient U-Net"(2021)

번호	관련증빙 명																
4	[소프트웨어] 드론촬영 데이터 정사영상생성을 위한 네트워크병렬처리 자동화시스템																
제 C-2021-000249 호  저작권 등록증 <table> <tr> <td>1. 저작물의 제호(명칭)</td><td>드론촬영 데이터 정사영상생성을 위한 네트워크 병렬처리 자동화시스템</td></tr> <tr> <td>2. 저작물의 종류</td><td>컴퓨터프로그램저작물>응용프로그램>산업용 S/W>지리정보시스템(GIS) S/W</td></tr> <tr> <td>3. 저작자 성명(법인명)</td><td>주식회사 엠지아이에스 경기도 광명시 새빛공원로</td></tr> <tr> <td>4. 생년월일(법인등록번호)</td><td>131111-0360427</td></tr> <tr> <td>5. 창작연월일</td><td>2020년12월22일</td></tr> <tr> <td>6. 공표연월일</td><td>-</td></tr> <tr> <td>7. 등록연월일</td><td>2021년01월04일</td></tr> <tr> <td>8. 등록사항</td><td>저작자 : 주식회사 엠지아이에스, 창작 : 2020.12.22</td></tr> </table> 「저작권법」 제53조에 따라 위와 같이 등록되었음을 증명합니다. 2021년 01월 04일 한국저작권위원회    		1. 저작물의 제호(명칭)	드론촬영 데이터 정사영상생성을 위한 네트워크 병렬처리 자동화시스템	2. 저작물의 종류	컴퓨터프로그램저작물>응용프로그램>산업용 S/W>지리정보시스템(GIS) S/W	3. 저작자 성명(법인명)	주식회사 엠지아이에스 경기도 광명시 새빛공원로	4. 생년월일(법인등록번호)	131111-0360427	5. 창작연월일	2020년12월22일	6. 공표연월일	-	7. 등록연월일	2021년01월04일	8. 등록사항	저작자 : 주식회사 엠지아이에스, 창작 : 2020.12.22
1. 저작물의 제호(명칭)	드론촬영 데이터 정사영상생성을 위한 네트워크 병렬처리 자동화시스템																
2. 저작물의 종류	컴퓨터프로그램저작물>응용프로그램>산업용 S/W>지리정보시스템(GIS) S/W																
3. 저작자 성명(법인명)	주식회사 엠지아이에스 경기도 광명시 새빛공원로																
4. 생년월일(법인등록번호)	131111-0360427																
5. 창작연월일	2020년12월22일																
6. 공표연월일	-																
7. 등록연월일	2021년01월04일																
8. 등록사항	저작자 : 주식회사 엠지아이에스, 창작 : 2020.12.22																

번호	관련증빙 명																
5	[소프트웨어] 3D 공간정보 웹 서비스																
제 C-2022-004301 호  저작권 등록증 <table> <tr> <td>1. 저작물의 제호(명칭)</td><td>3D 공간정보 웹 서비스</td></tr> <tr> <td>2. 저작물의 종류</td><td>컴퓨터프로그램저작물>응용프로그램>프로그램 개발용 S/W>웹 개발 도구</td></tr> <tr> <td>3. 저작자 성명(법인명)</td><td>주식회사 엠지아이에스 경기도 광명시 새빛공원로</td></tr> <tr> <td>4. 생년월일(법인등록번호)</td><td>131111-0360427</td></tr> <tr> <td>5. 창작연월일</td><td>2021년12월31일</td></tr> <tr> <td>6. 공표연월일</td><td>-</td></tr> <tr> <td>7. 등록연월일</td><td>2022년01월10일</td></tr> <tr> <td>8. 등록사항</td><td>저작자 : 주식회사 엠지아이에스, 창작 : 2021.12.31</td></tr> </table> 「저작권법」 제53조에 따라 위와 같이 등록되었음을 증명합니다. 2022년 01월 10일 한국저작권위원회    		1. 저작물의 제호(명칭)	3D 공간정보 웹 서비스	2. 저작물의 종류	컴퓨터프로그램저작물>응용프로그램>프로그램 개발용 S/W>웹 개발 도구	3. 저작자 성명(법인명)	주식회사 엠지아이에스 경기도 광명시 새빛공원로	4. 생년월일(법인등록번호)	131111-0360427	5. 창작연월일	2021년12월31일	6. 공표연월일	-	7. 등록연월일	2022년01월10일	8. 등록사항	저작자 : 주식회사 엠지아이에스, 창작 : 2021.12.31
1. 저작물의 제호(명칭)	3D 공간정보 웹 서비스																
2. 저작물의 종류	컴퓨터프로그램저작물>응용프로그램>프로그램 개발용 S/W>웹 개발 도구																
3. 저작자 성명(법인명)	주식회사 엠지아이에스 경기도 광명시 새빛공원로																
4. 생년월일(법인등록번호)	131111-0360427																
5. 창작연월일	2021년12월31일																
6. 공표연월일	-																
7. 등록연월일	2022년01월10일																
8. 등록사항	저작자 : 주식회사 엠지아이에스, 창작 : 2021.12.31																

번호	관련증빙 명		
6	[시제품] K KU-PPS(CPU 병렬처리 시스템)		

[서식]

「시제품 제작」 성과 확인서

☐ 과제정보

사업명	국토교통기술촉진사업	과제번호	20CTAP-C158040-01
과제명	3차원 공간정보 제작을 위한 CPU 병렬처리 시스템과 실시간으로 변화를 갱신할 수 있는 Local Dynamic Mapping 소프트웨어 개발		
연구기관	건국대학교 산학협력단	연구책임자	서정현

☐ 현장시험·검증 성과

구분	시험시공		
시제품명	K KU-PPS		
제작일시	2020. 11. 18	제작업체	컴퓨터안심서비스/ 분당데이터
목적	3차원 공간정보 생성시간 단축		
내용	3차원 공간정보 생성시간 10시간 이내 단축		
활용계획	사진 매수가 많은 대규모 지역 3D-mapping		
증빙	[첨부] 증빙 내역(시험결과보고서)		

위 연구과제를 통해 발생된 성과와 증빙이 상기와 같음을 확인합니다.

2020 년 12 월 9 일

연구기관

건국대학교 산학협력단

책임자

송창선



(기관인)

국토교통부장관 귀하

시제품 시험결과 보고서

< KCU-PPS 성능평가 >

2020년 12월 9일

□ 기본정보

실험명	KKU-PPS 성능평가	수행기간	2020. 11. 30 - 2020. 12. 08
사업명	국토교통기술촉진사업	과제번호	20CTAP-C158040-01
과제명	3차원 공간정보 제작을 위한 CPU 병렬처리 시스템과 실시간으로 변화를 갱신할 수 있는 Local Dynamic Mapping 소프트웨어 개발		
수행기관	기관명	건국대학교	
	주소	서울시 광진구 능동로 120 건국대학교	
실험책임자	설명	서 정 현	TEL 010-9845-6222
	E-mail	a21689@konkuk.ac.kr	FAX -

□ 실험개요

- 실험목적
 - Stand alone(1CPU)과 CPU 병렬처리시스템 (KKU-PPS)의 대규모지역 3차원 공간정보 처리속도 비교
- 실험내용 및 방법
 - 자료처리조건
 1. CPU: Ryzen 9 3900X, RAM: 64Gbyte, GPU: 2070SUPER
 2. 대상면적: 0.65km², 사진매수: 6,700매
 3. 처리option: 정확도 2cm
 - 방법: Server(PC 1대)+Client(PC 1대)+ Node(PC 7대)로 분산병렬처리시스템 구성하여 Stand alone(PC 1대)와의 전체 최종산출물 처리시간 비교



KKU-PPS



최종산출물예시(Orthophoto)

□ 실험결과물

- 산출물: 3차원 공간정보(DSM, orthophoto 등), 최종 산출물 처리시간

□ 실험결과

1. Stand alone Workflow

1. Align Photos - Accuracy: High, Preselection: Generic + Reference(Source)	11h 30m
2. Optimize Camera	22m
3. Build Dense Cloud - Quality: High, Filter: Aggressive	4d 16h 9m
5. Build Mesh - Quality: High, Source: Depth Map	4d 12h 35m
6. Build DEM(DSM) -Source: Mesh	24m
7. Build Orthomosaic	6h 30m

Total: 10d 19h 30m

2. KCU-PPS Workflow

1. Align Photos - Accuracy: High, Preselection: Generic + Reference(Source)	2h 24m
2. Optimize Camera	11m
3. Build Dense Cloud - Quality: High, Filter: Aggressive	6h 38m
5. Build Mesh - Quality: High, Source: Depth Map	8h 2m
6. Build DEM(DSM) -Source: Mesh	13m
7. Build Orthomosaic	6h 20m

Total: 23h 48m

3. 결과비교

시험목적	대상품목	처리시간	비고
3차원공간정보 생성	Stand alone	10d 19h 30m	-
	KCU-PPS	23h 48m	-

- 실험 결과 대규모지역에 대한 3차원 공간정보 생성 처리시간 1/10 단축 확인.

2020년 12월 9일

연구 책임자: 서 정 현



번호	관련증빙 명
7	[특허출원] 3차원 공간정보 생성을 위한 병렬처리 장치 및 방법

2020. 12. 15.

출원번호통지서

관 인 생 략

출 원 번 호 통 지 서

출 원 일 자 2020.12.14
특 기 사 항 심사청구(유) 공개신청(무) 참조번호(KU20L011P)
출 원 번 호 10-2020-0174628 (접수번호 1-1-2020-1355995-25)
(DAS접근코드 40C4)
출 원 인 명 칭 건국대학교 산학협력단(2-2004-015764-8)
대 리 인 성 명 정부연(9-2007-001006-1)
발 명 자 성 명 김상호
발 명 의 명 칭 3차원 공간정보 생성을 위한 병렬처리 장치 및 방법

특 허 청 장

<< 안내 >>

1. 귀하의 출원은 위와 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 통해 확인하실 수 있습니다.
2. 출원에 따른 수수료는 접수일로부터 다음날까지 통보된 납입영수증에 성명, 납부자번호 등을 기재하여 가까운 우체국 또는 은행에 납부하여야 합니다.
※ 납부자번호 : 0131(기관코드) + 접수번호
3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고객번호 정보변경(경정), 정정신고서]를 제출하여야 출원 이후의 각종 통지서를 정상적으로 받을 수 있습니다.
※ 특허로(patent.go.kr) 접속 > 민원서식다운로드 > 특허법 시행규칙 별지 제5호 서식
4. 특허(실용신안등록)출원은 명세서 또는 도면의 보정이 필요한 경우, 등록결정 이전 또는 의견서 제출기간 이내에 출원서에 최초로 첨부된 명세서 또는 도면에 기재된 사항의 범위 안에서 보정할 수 있습니다.
5. 외국으로 출원하고자 하는 경우 PCT 제도(특허·실용신안)나 마드리드 제도(상표)를 이용할 수 있습니다. 국내출원일을 외국에서 인정받고자 하는 경우에는 국내출원일로부터 일정한 기간 내에 외국에 출원하여야 우선권을 인정받을 수 있습니다.
※ 제도 안내 : <http://www.kipo.go.kr-특허마당-PCT/마드리드>
※ 우선권 인정기간 : 특허·실용신안은 12개월, 상표·디자인은 6개월 이내
※ 미국특허상표청의 선출원을 기초로 우리나라에 우선권주장출원 시, 선출원이 미공개상태이면, 우선일

<https://www.patent.go.kr/jsp/kiponet/ir/receipt/online/applNoOfficAct.do>
1/4

- 15 -

【서지사항】

【서류명】	특허출원서
【참조번호】	KU20L011P
【출원구분】	특허출원
【출원인】	
【명칭】	건국대학교 산학협력단
【특허고객번호】	2-2004-015764-8
【대리인】	
【성명】	정부연
【대리인번호】	9-2007-001006-1
【포괄위임등록번호】	2018-011491-1
【발명의 국문명칭】	3차원 공간정보 생성을 위한 병렬처리 장치 및 방법
【발명의 영문명칭】	PARALLEL PROCESSING APPARATUS AND METHOD FOR GENERATING 3D SPATIAL INFORMATION
【발명자】	
【성명】	김상호
【성명의 영문표기】	KIM Sang Ho
【주민등록번호】	660418-1XXXXXX
【우편번호】	06012
【주소】	서울특별시 강남구 도산대로89길 35 청담병신칼릭스빌2차 501호
【출원언어】	국어
【심사청구】	청구

【공지에외적용대상증명서류의 내용】

【공개형태】 논문공개
【공개일자】 2020.06.12

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 1615011585
【과제번호】 158040
【부처명】 국토교통부
【과제관리(전문)기관명】 국토교통과학기술진흥원

【연구사업명】 국토교통기술촉진연구사업

【연구과제명】 3차원 공간정보 제작을 위한 CPU 병렬처리 시스템과 실시간
 으로 변화를 갱신할 수 있는 Local Dynamic Mapping 소프
 트웨어 개발

【기여율】 1/1

【과제수행기관명】 건국대학교산학협력단

【연구기간】 2020.04.13 - 2020.12.31

【취지】 위와 같이 특허청장에게 제출합니다.

대리인 정부연

(서명 또는 인)

【수수료】

【출원료】 0 면 46,000 원

【가산출원료】 35 면 0 원

【우선권주장료】 0 건 0 원

【요약서】

【요약】

본 발명은 3차원 공간정보 생성을 위한 병렬처리 장치 및 방법에 관한 것으로, 상기 장치는 적어도 하나의 프로세서로 각각 구성된 복수의 노드들; 주어진 경로를 따라 이동하면서 특정 주기동안 해당 위치에서의 주변 공간을 촬영하여 구성된 복수의 단위 공간영상들을 시계열 순으로 수신하는 공간영상 수신부; 상기 복수의 단위 공간영상들의 개수를 기초로 상기 복수의 노드들 중 최적의 병렬처리를 위한 노드들을 결정하는 병렬처리 노드 결정부; 및 상기 결정된 노드들에 상기 복수의 단위 공간영상들을 분배하고 상기 결정된 노드들 중 하나가 상기 복수의 단위 공간영상들에서 중첩된 공간영역을 검출하도록 하는 병렬처리 노드 제어부를 포함한다.

【대표도】

도 5

번호	관련증빙 명
8	[특허출원] 네트워크 병렬 처리를 이용한 정사영상 자동화 생성 방법

관인생략

출원번호통지서

출원일자 2021.02.05

특기사항 심사청구(유) 공개신청(무)

출원번호 10-2021-0016810 (접수번호 1-1-2021-0152323-25)
(DAS접근코드631B)

출원인명칭 주식회사 엠지아이에스(1-2020-097594-3)

대리인성명 최영복(9-1998-000571-2)

발명자성명 박정임 김정우

발명의명칭 네트워크 병렬 처리를 이용한 정사영상 자동화 생성 방법

특허청장

<< 안내 >>

1. 귀하의 출원은 위와 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 이용하여 특허로 홈페이지(www.patent.go.kr)에서 확인하실 수 있습니다.
2. 출원에 따른 수수료는 접수일로부터 다음날까지 동봉된 납입영수증에 성명, 납부자번호 등을 기재하여 가까운 은행 또는 우체국에 납부하여야 합니다.
※ 납부자번호 : 0131(기관코드) + 접수번호
3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고객번호 정보변경(경정), 정정신고서]를 제출하여야 출원 이후의 각종 통지서를 정상적으로 받을 수 있습니다.
4. 기타 심사 절차(제도)에 관한 사항은 특허청 홈페이지를 참고하시거나 특허고객상담센터(☎ 1544-8080)에 문의하여 주시기 바랍니다.
※ 심사제도 안내 : <http://www.kipo.go.kr>-지식재산제도

【서지사항】

【서류명】 특허출원서
【출원구분】 특허출원
【출원인】
【명칭】 주식회사 엠지아이에스
【특허고객번호】 1-2020-097594-3
【대리인】
【성명】 최영복
【대리인번호】 9-1998-000571-2
【포괄위임등록번호】 2021-000380-6
【발명의 국문영칭】 네트워크 병렬 처리를 이용한 정사영상 자동화 생성 방법
【발명의 영문영칭】 Ortho image automation generating method of using network parallel processing
【발명자】
【성명】 박정임
【성명의 영문표기】 PARK, JUNG IM
【주민등록번호】 *****
【우편번호】 07993
【주소】 서울특별시 양천구 목동서로 234, 102동 703호(목동, 성원 아파트)
【발명자】
【성명】 김정우
【성명의 영문표기】 KIM, JUNG WOO

2021-02-05

【주민등록번호】 ★★★★★-★★★★★

【우편번호】 14988

【주소】 경기도 시흥시 목감중앙로 45, 1110동 1903호(조남동, 리벤티움)

【출원언어】 국어

【심사청구】 청구

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 1615011585

【과제번호】 158040

【부처명】 국토교통부

【과제관리(전문)기관명】 국토교통과학기술진흥원

【연구사업명】 국토교통기술혁신연구(R&D)

【연구과제명】 3차원 공간정보 제작을 위한 CPU 병렬처리 시스템과 실시간으로 변화를 갱신할 수 있는 Local Dynamic Mapping 소프트웨어 개발

【기여율】 1/2, 1/2

【과제수행기관명】 건국대학교산학협력단, 주식회사 엠지아이에스

【연구기간】 2020.04.13 - 2020.12.31

【취지】 위와 같이 특허청장에게 제출합니다.

대리인 최영복 (서명 또는 인)

【수수료】

【요약서】**【요약】**

본 발명은 네트워크 병렬 처리를 이용한 정사영상 자동화 생성 방법을 개시한다. 이러한 본 발명은 네트워크 병렬 처리의 세팅과 작동에 대한 정사영상 생성 공정의 다소 복잡하고 번거로운 작동 방법을 자동화 불로서 제이하면서 드론에 의해 촬영되는 대량의 데이터들의 정사영상 생성 공정을 자동화시킨 것이고, 이에 따라 대량의 사진 데이터를 통한 정사 영상 생성 시간을 단축하면서 누구나 손쉽게 정사 영상을 사용할 수 있도록 함은 물론, 정사 영상 생성 공정에서의 단일 프로그램 사용에 따른 불편함과 어려움을 개선하는 것이다.

【대표도】**도 2**

번호	관련증빙 명
9	Local Dynamic Mapping S/W
Software Source Code [Main] <pre> # Directory initialize path_list = [path_ort,path_drp,path_kpt,path_ojp,path_mtc] pkgd.Dir_init(path_list) # Data import print('[Import Data]') ortho_img, drone_img, geotransfrom, epsg = pkgd.import_data(source_or, source_dr, path_ort, path_drp,path_ojp) # Matching image print('[Start to matching]') pkgd.Matching(ortho_img, drone_img, path_kpt, path_mtc) # Georeferencing print('[Start to Georeferencing]') pkgd.Georef(path_mtc, geotransfrom, epsg, path_out) </pre> [Data Import Module] <pre> def import_data(source_or, source_dr, path_ort, path_drp, path_ojp): # Copy orthomosaic-map and drone-photo ortho_mv = shutil.copy(source_or, path_ort) drone_mv = shutil.copy(source_dr, path_drp) # import orthomosaic map ortho_list = os.listdir(path_ort) ortho_name = ortho_list[-1] ortho = gdal.Open(os.path.join(path_ort+ortho_name)) ortho_arr = ortho.ReadAsArray() print(' Orthomosaic map: ' + ortho_name) print(f" - Format: {ortho.GetDriver().ShortName}/{ortho.GetDriver().LongName}") print(f" - Size: {ortho.RasterXSize} x {ortho.RasterYSize} x {ortho.RasterCount}") Projection = ortho.GetProjection() #print("Projection:", Projection) epsg = Projection[-7:-3] epsg = int(eps) print(' - EPSG:', epsg) rows = ortho.RasterYSize cols = ortho.RasterXSize x0, dx, dxdy, y0, dydx, dy = ortho.GetGeoTransform() geotransfrom = ortho.GetGeoTransform() x1 = x0 + dx * cols y1 = y0 + dy * rows new_dx = round(dx,3) print(f" - Range: {x0}:{x1}, {y0}:{y1}") print(f" - Ground sample distance : {new_dx}m") print('') ortho_arr = ortho_arr.transpose(1,2,0) cv2.imwrite(path_ojp+ 'Ortho.jpg', ortho_arr) ortho_img = cv2.imread(path_ojp+ 'Ortho.jpg') #ortho_img = cv2.imread(os.path.join(path_ort+ortho_name)) # import drone_image dron_list = os.listdir(path_drp) dron_name = dron_list[-1] drone_img = cv2.imread(path_drp+dron_name) # Drone image coordinate drone_img_path = os.path.join(path_drp+dron_name) def get_exif(filename): image = Image.open(filename) image.verify() return image._getexif() def get_geotagging(exif): if not exif: raise ValueError("No EXIF metadata found") geotagging = {} for (idx, tag) in TAGS.items(): if tag == 'GPSInfo': if idx not in exif: raise ValueError("No EXIF geotagging found") for (key, val) in GPSTAGS.items(): if key in exif[idx]: geotagging[val] = exif[idx][key] return geotagging </pre>	

```

exif = get_exif(drone_img_path)
geotags = get_geotagging(exif)

print(' Drone Image Coordinate')
print(' - Latitude:', geotags['GPSLatitude'])
print(' - Longitude:', geotags['GPSLongitude'])
print(' - Altitude:', geotags['GPSAltitude'])
print('')
return ortho_img, drone_img, geotransfrom, epsg

```

[Image Matching Module]

```

def Matching(ortho_img, drone_img, path_kpt, path_mtc):
    # Keypoint
    ortho = ortho_img
    ortho_bi = cv2.bilateralFilter(ortho, 9, 150, 50)
    ortho_gray = cv2.cvtColor(ortho_bi, cv2.COLOR_BGR2GRAY)

    drone = drone_img
    drone_bi = cv2.bilateralFilter(drone, 9, 150, 50)
    drone_gray = cv2.cvtColor(drone_bi, cv2.COLOR_BGR2GRAY)
    print(' Start to detect keypoints...')

    # SIFT
    detector = cv2.SIFT_create()
    kp1, des1 = detector.detectAndCompute(ortho_gray, None)
    kp2, des2 = detector.detectAndCompute(drone_gray, None)
    print(' - Orhtomosaic keypoints:', len(kp1))
    print(' - Drone_image keypoints:', len(kp2))

    dst1 = cv2.drawKeypoints(ortho, kp1, None, color=(0,0,255))
    dst2 = cv2.drawKeypoints(drone, kp2, None, color=(0,0,255))
    cv2.imwrite(path_kpt+'ortho_kp.jpg', dst1)
    cv2.imwrite(path_kpt+'drone_kp.jpg', dst2)

    # BFMatcher
    print(' Matching key points...')
    match = cv2.BFMatcher()
    matches = match.knnMatch(des1, des2, k=2)

    good = []
    for m,n in matches:
        #Distance parameter
        #0.485
        if m.distance < 0.485*n.distance:
            good.append(m)

    draw_params = dict(matchColor = (0,0,255), singlePointColor = None, flags = 2)
    Match_image = cv2.drawMatches(ortho_gray, kp1, drone_gray, kp2, good, None, **draw_params)
    print(' - Filtered tie points:', len(good))
    print('')
    cv2.imwrite(path_mtc+"DrawMatches.jpg", Match_image)

    # Perspective Transform
    MIN_MATCH_COUNT = 10

    if len(good) > MIN_MATCH_COUNT:
        ortho_pts = np.float32([ kp1[m.queryIdx].pt for m in good]).reshape(-1,1,2)
        drone_pts = np.float32([ kp2[m.trainIdx].pt for m in good]).reshape(-1,1,2)
        "change ransac parameter"
        M, mask = cv2.findHomography(drone_pts, ortho_pts, cv2.RANSAC, 5.0)
        h,w = ortho_gray.shape
        pts = np.float32([ [0,0], [0,h-1], [w-1,h-1], [w-1,0] ]).reshape(-1,1,2)
        dst = cv2.perspectiveTransform(pts, M)
        Overlapping = cv2.polyline(drone_gray, [np.int32(dst)], True, (0,0,255), 20, cv2.LINE_AA)
        cv2.imwrite(path_mtc+"Overlapping.jpg", Overlapping)
    else:
        print("Not enough matches are found - %d/%d", (len(good)/MIN_MATCH_COUNT))

    Perspectives = cv2.warpPerspective(drone, M, (ortho.shape[1], ortho.shape[0]))
    Output= cv2.subtract(ortho, Perspectives)
    Output= cv2.subtract(Output, Perspectives)
    Output= cv2.add(Output, Perspectives)

    cv2.imwrite(path_mtc+"Matching_Image.tif", Output)

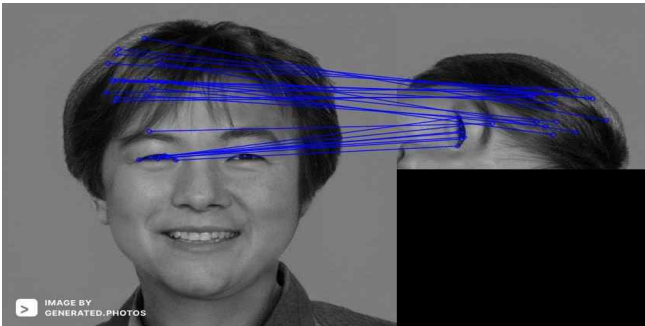
```

[Georeferencing Module]

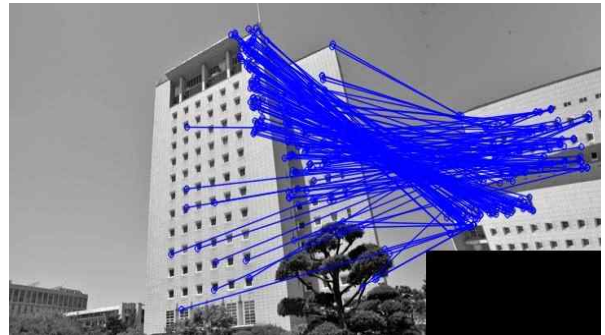
```
def Georef(path_mtc, geotransfrom, epsg, path_out):  
    print(' - Georeferencing...')  
  
    Matched_img = os.path.join(path_mtc+'Matching_Image.tif')  
    dst_filename = os.path.join(path_out+'Image_matched.tif')  
  
    src_ds = gdal.Open(Matched_img)  
    format = "GTiff"  
    driver = gdal.GetDriverByName(format)  
    dst_ds = driver.CreateCopy(dst_filename, src_ds, 0)  
  
    Geotransform = geotransfrom  
    dst_ds.SetGeoTransform(Geotransform)  
  
    Epsg = epsg  
    srs = osr.SpatialReference()  
    srs.ImportFromEPSG(Epsg)  
    dest_wkt = srs.ExportToWkt()  
    dst_ds.SetProjection(dest_wkt)  
  
    print(' - Processing Finished')  
    dst_ds = None  
    src_ds = None
```

Tie point 검출

[사진의 중심좌표와 기울기가 없는 일반사진 Pair의 Tie point 검출]

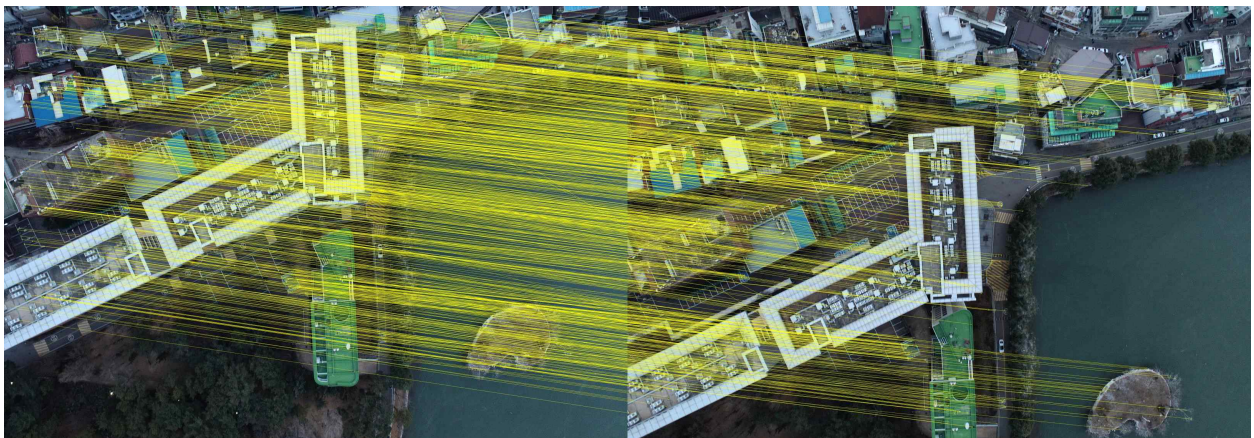


Tie point 18개 검출



Tie point 163개 검출

[사진의 중심좌표와 기울기가 있는 드론 촬영사진 Pair의 Tie point 검출]



Tie point 1,176개 검출

번호	관련증빙 명
10	Local Dynamic Mapping S/W
Software Source Code [Main] <pre> def Main_ldm(source, path_dst): source = source path_dst = path_dst target_size = 1024 BATCH_SIZE = 1 path_list = [path_src,path_ojp, path_sli, path_tls,path_sth, path_hcn, path_ppr, path_pol] pkgd.Dir_init(path_list, path_out, path_dst) print('< Processing has began. >') pkgd.filemv(source, path_src) # Data import - Geotiff파일 정보 및 이미지 slicing x0, y0, scale_factor, _, OrthoXsize, OrthoYsize, image_name, projection = pkgd.Image_slice(path_src, path_ojp, path_sli, target_size) scale_factor = round(scale_factor, 4) gpus = tf.config.experimental.list_physical_devices('GPU') if gpus: try: # Currently, memory growth needs to be the same across GPUs for gpu in gpus: tf.config.experimental.set_memory_growth(gpu, True) logical_gpus = tf.config.experimental.list_logical_devices('GPU') print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs") except RuntimeError as e: # Memory growth must be set before GPUs have been initialized print(e) print('') print('[Step 3] Image Translation.') Translate_dataset = tf.data.Dataset.list_files(path_sli+'*.jpg', shuffle=False) Translate_dataset = Translate_dataset.map(pkgd.load_image_src) Translate_dataset = Translate_dataset.batch(BATCH_SIZE) print(' - Checking the data...') print(' - Loading Model...') generator = tf.keras.models.load_model(path_model+'Model_48.h5') file_list = os.listdir(path_sli,) count = [] print(' - Translating has started ,') for img in Translate_dataset: prediction_img = pkgd.Translate_images(generator, img) file_name = file_list[len(count)] img_name = (file_name) cv2.imwrite(path_tls+img_name, prediction_img) if len(count) % 28 == 0: print(" Completed : ({} / {})".format(len(count), len(file_list))) count.append(0) print(" Completed : ({} / {})".format(len(file_list), len(file_list))) print(' - Translating has completed') # Image Stitching Module pkgd.Stitching(path_src, path_tls, path_hcn, path_sth,target_size) # Vector Extract pkgd.Extraction_shp(path_sth, path_ppr,path_pol,path_out, OrthoXsize, OrthoYsize, scale_factor, x0, y0, image_name, projection) #path_list = [path_src,path_ojp, path_sli, path_tls, path_hcn, path_ppr, path_pol] #pkgd.Dir_init(path_list, path_out, path_dst) print('') print('Processing Finished, Please Check the Files') Main_ldm(source, path_dst) </pre>	

[Preprocessing Module]

```
def Image_slice(path_src, path_ojp, path_sli, target_size):
    print('')
    print('[Step 2] Importing Orthomosaic Map')
    file_list = os.listdir(path_src)
    file_name = file_list[-1]
    image_name = file_name[:-4]

    # Orthomosaic Data
    Ortho = gdal.Open(path_src + file_name)
    Ortho_arr = Ortho.ReadAsArray()
    print(' - File Name: ' + file_name)
    print(f" - Format: {Ortho.GetDriver().ShortName}/{Ortho.GetDriver().LongName}")
    print(f" - Size: {Ortho.RasterXSize} x {Ortho.RasterYSize} x {Ortho.RasterCount}")
    Projection = Ortho.GetProjection()
    Projection_val_1 = Projection[-14:-10]
    Projection_val_2 = Projection[-7:-3]
    projection = Projection_val_1 + ':' + Projection_val_2
    print(" - Projection:", projection)
    #print(" - Projection:", Projection)
    rows = Ortho.RasterYSize
    cols = Ortho.RasterXSize
    x0, dx, dxdy, y0, dydx, dy = Ortho.GetGeoTransform()
    x1 = x0 + dx * cols
    y1 = y0 + dy * rows
    print(f" - Range: {x0}:{x1}, {y0}:{y1}")
    print(f" - Ground sample distance : {dx}")

    x0 = round(x0)
    y0 = round(y0)

    Ortho_arr = Ortho_arr.transpose(1,2,0)
    cv2.imwrite(path_ojp+ 'Ortho.jpg', Ortho_arr)

    ## CLAHE
    cla = cv2.imread(path_ojp+'Ortho.jpg')
    img_yuv = cv2.cvtColor(cla, cv2.COLOR_BGR2YUV)
    img_clahe = img_yuv.copy()
    clahe = cv2.createCLAHE(clipLimit=1.25, tileGridSize=(4,4))
    img_clahe[:, :, 0] = clahe.apply(img_clahe[:, :, 0])
    img_clahe = cv2.cvtColor(img_clahe, cv2.COLOR_YUV2BGR)

    Ortho_jpg = gdal.Open(path_ojp + 'Ortho.jpg')
    arr = Ortho_jpg.ReadAsArray()
    arr = arr.transpose(1,2,0)

    ## Image slicing
    print(' - Slicing has started')
    src_h = arr.shape[0]
    src_w = arr.shape[1]
    last_h = src_h-target_size
    last_w = src_w-target_size
    divide_h = math.ceil(src_h/target_size)
    divide_w = math.ceil(src_w/target_size)
    y = 0
    x = 0
    name_counter = 1

    for i in range(divide_h):
        if i < (divide_h-1):
            for j in range(divide_w):
                if j < (divide_w-1):
                    h = y*target_size
                    w = x*target_size
                    img_trim = arr[h:h+target_size, w:w+target_size]
                    imgname = (str(name_counter) + ".jpg")
                    cv2.imwrite(os.path.join(path_sli, imgname), img_trim)
                    #print('Image saved: ', imgname)
                    x += 1
                    name_counter += 1

                if j == (divide_w-1):
                    h = y*target_size
                    img_trim = arr[h:h+target_size, last_w:src_w]
                    imgname = (str(name_counter) + ".jpg")
                    cv2.imwrite(os.path.join(path_sli, imgname), img_trim)
                    #print('Image saved: ', imgname)
                    x = 0
                    y += 1
                    name_counter += 1
```

```

    if i == (divide_h-1):
        for j in range(divide_w):
            if j < (divide_w-1):
                h = y*target_size
                w = x*target_size
                img_trim = arr[last_h:last_h+target_size, w:w+target_size]
                imgname = (str(name_counter) + ".jpg")
                cv2.imwrite(os.path.join(path_sli,imgname),img_trim)
                #print('Image saved: ', imgname)
                x += 1
                name_counter += 1

            if j == (divide_w-1):
                img_trim = arr[last_h:src_h, last_w:src_w]
                imgname = (str(name_counter) + ".jpg")
                cv2.imwrite(os.path.join(path_sli,imgname),img_trim)
                #print('Image saved: ', imgname)
                x = 0
                y = 0
                name_counter += 1
print(' - Slicing has completed')
return x0,y0, dx, dy, cols, rows, image_name, projection

```

[Image to Image translation Module]

```

def load_src(image_file):
    image = tf.io.read_file(image_file)
    input_image = tf.image.decode_jpeg(image)
    input_image = tf.cast(input_image, tf.float32)
    return input_image

def resize_src(input_image, height, width):
    input_image = tf.image.resize(input_image, [height, width],
                                   method = tf.image.ResizeMethod.NEAREST_NEIGHBOR)
    return input_image

def normalize_src(input_image):
    input_image = (input_image/127.5) -1
    return input_image

def load_image_src( image_file):
    input_image = load_src(image_file)
    input_image = resize_src(input_image, IMG_HEIGHT, IMG_WIDTH)
    input_image = normalize_src(input_image)
    return input_image

def Translate_images(model, img_input):
    prediction = model(img_input, training = False)
    prediction_np = prediction.numpy()
    prediction_np = np.squeeze(prediction_np)
    prediction_img = (prediction_np+1)*127.5
    return prediction_img

def downsample(filters, size, apply_batchnorm = True):
    initializer = tf.random_normal_initializer(0., 0.02)
    result = tf.keras.Sequential()
    result.add(tf.keras.layers.Conv2D(filters, size, strides=2, padding = 'same',
                                       kernel_initializer=initializer, use_bias=False))
    if apply_batchnorm:
        result.add(tf.keras.layers.BatchNormalization())
    result.add(tf.keras.layers.LeakyReLU())
    return result

def upsample(filters, size, apply_dropout=False):
    initializer = tf.random_normal_initializer(0., 0.02)

    result = tf.keras.Sequential()
    result.add(tf.keras.layers.Conv2DTranspose(filters, size, strides=2, padding='same',
                                                kernel_initializer=initializer, use_bias=False))
    result.add(tf.keras.layers.BatchNormalization())
    if apply_dropout:
        result.add(tf.keras.layers.Dropout(0.5))
    result.add(tf.keras.layers.ReLU())
    return result

```

```

def Generator():
    inputs = tf.keras.layers.Input(shape=[1024,1024,3])
    down_stack = [
        downsample(64, 4, apply_batchnorm=False),
        downsample(128, 4),
        downsample(256, 4),
        downsample(512, 4),
        downsample(512, 4),
        downsample(512, 4),
        downsample(512, 4),
        downsample(512, 4),
    ]

    up_stack= [
        upsample(512, 4, apply_dropout=True),
        upsample(512, 4, apply_dropout=True),
        upsample(512, 4, apply_dropout=True),
        upsample(512, 4),
        upsample(256, 4),
        upsample(128, 4),
        upsample(64, 4),
    ]

    initializer = tf.random_normal_initializer(0., 0.02)
    last = tf.keras.layers.Conv2DTranspose(OUTPUT_CHANNELS, 4, strides=2, padding='same',
                                           kernel_initializer=initializer, activation='tanh')

    skips = []
    for down in down_stack:
        x = down(x)
        skips.append(x)

    skips = reversed(skips[:-1])
    for up, skip in zip(up_stack, skips):
        x = up(x)
        x = tf.keras.layers.Concatenate()([x, skip])
    x = last(x)
    return tf.keras.Model(inputs=inputs, outputs=x)

def Discriminator():
    initializer = tf.random_normal_initializer(0.,0.02)
    inp = tf.keras.layers.Input(shape=[1024,1024,3], name = 'input_image')
    tar = tf.keras.layers.Input(shape=[1024,1024,3], name = 'target_image')

    x = tf.keras.layers.concatenate([inp, tar])

    down1 = downsample(64,4,False)(x)
    down2 = downsample(128,4)(down1)
    down3 = downsample(256,4)(down2)

    zero_pad1 = tf.keras.layers.ZeroPadding2D()(down3)
    conv = tf.keras.layers.Conv2D(512,4,strides=1, kernel_initializer=initializer, use_bias = False)(zero_pad1)
    batchnorm1 = tf.keras.layers.BatchNormalization()(conv)
    leaky_relu = tf.keras.layers.LeakyReLU()(batchnorm1)
    zero_pad2 = tf.keras.layers.ZeroPadding2D()(leaky_relu)
    last = tf.keras.layers.Conv2D(1,4,strides=1, kernel_initializer=initializer, use_bias = False)(zero_pad2)
    return tf.keras.Model(inputs=[inp, tar], outputs=last)

```

[Postprocessing & Polygonization Module]

```

def Stitching(path_src, path_tls, path_hcn, path_sth,target_size):
    ### Check_size
    print('')
    print('[Step 4] Vector Extraction.')
    def Check_size(path_src, target_size):
        #print(" - Stitching has started")
        file_list = os.listdir(path_src)
        image_name = file_list[-1]
        src = gdal.Open(path_src + image_name)
        arr = src.ReadAsArray()
        arr = arr.transpose(1,2,0)
        #print(arr.shape)
        src_h = arr.shape[0]
        src_w = arr.shape[1]
        divide_h = math.ceil(src_h/target_size)
        divide_w = math.ceil(src_w/target_size)
        crop_w_point = ((src_w)-(divide_w-1)*target_size)
        crop_h_point = ((src_h)-(divide_h-1)*target_size)
        #print('divide_h:', divide_h, 'divide_w:', divide_w)
        return divide_h, divide_w, crop_w_point, crop_h_point, image_name

    divide_h, divide_w, crop_w_point, crop_h_point, image_name = Check_size(path_src, target_size)

```

```

def List_Dist(path_tls, divide_w):
    sli_list = os.listdir(path_tls)
    num_of_images = len(sli_list)
    list_n = []
    list_w = []

    for a in range(num_of_images):
        if (a+1) % (divide_w) != 0:
            list_n.append(a+1)
        else :
            list_w.append(a+1)
    #print(len(list_n))
    #print(len(list_w))
    return list_n, list_w

```

```
list_n, list_w = List_Dist(path_tls, divide_w)
```

```

### Hconcat
def Hconcat(path_tls, divide_w, list_n, list_w, path_hcn, crop_w_point):
    img_list_n = []
    #img_list_w = []
    count_s = 0

    for img in list_n:
        m = cv2.imread(path_tls + str(img)+ '.jpg')
        img_list_n.append(m)

        if len(img_list_n) == (divide_w-1):
            nf = str(list_w[count_s])+'.jpg'
            n = cv2.imread(path_tls + nf)
            n = n[:, (target_size-crop_w_point):target_size]
            img_list_n.append(n)
            #print(n.shape)

            h_concat = cv2.hconcat(img_list_n)
            hcon_name = (str(count_s+1))+'.jpg'
            cv2.imwrite(os.path.join(path_hcn, hcon_name),h_concat)
            #print("Saved: ",hcon_name)
            img_list_n =[]
            count_s += 1

```

```
Hconcat(path_tls, divide_w, list_n, list_w, path_hcn, crop_w_point)
```

```

def Vconcat(path_hcn, divide_h, target_size, crop_h_point, path_sth, image_name):
    vcon_list = os.listdir(path_hcn)
    Upper_img = []
    Lower_img = []

    for a in range(len(vcon_list)):
        if (a+1) % (divide_h) != 0:
            Upper_img.append(a+1)
        if (a+1) % (divide_h) == 0:
            Lower_img.append(a+1)
    #print(len(Upper_img))
    #print(len(Lower_img))

    Vconcat_img = []

    for img in Upper_img:
        m = cv2.imread(path_hcn+ str(img)+'.jpg')
        Vconcat_img.append(m)

        if len(Vconcat_img) == (divide_h-1):
            n = cv2.imread(path_hcn + str(Lower_img[0])+'.jpg')
            n = n[(target_size-crop_h_point):target_size,:]
            Vconcat_img.append(n)
            Stitched = cv2.vconcat(Vconcat_img)
            Stitched_name = (image_name[:-4] + "_translated" + ".jpg")
            cv2.imwrite(os.path.join(path_sth, Stitched_name), Stitched)
            print(" - Stitching has completed")

```

```
Vconcat(path_hcn, divide_h, target_size, crop_h_point, path_sth, image_name)
```

```

def Extraction_shp(path_sth, path_ppr,path_pol,path_out, OrthoXsize, OrthoYsize, scale_factor,
                                                            x0, y0, image_name, projection):
    def contours(path_sth,path_ppr, OrthoXsize, OrthoYsize, scale_factor):
        Sttched_list = os.listdir(path_sth)
        Sttched_image = Sttched_list[-1]
        #print(Sttched_image)
        Sttched = cv2.imread(os.path.join(path_sth+Sttched_image),cv2.IMREAD_GRAYSCALE)
        #print(Sttched.shape)

        # Preprocessing
        kernel = cv2.getStructuringElement(shape = cv2.MORPH_RECT, ksize= (3,3))
        erode = cv2.erode(Sttched, kernel,iterations =3)
        dilate = cv2.dilate(erode, kernel,iterations =3)
        cv2.imwrite(os.path.join(path_ppr,'opening.jpg'), dilate)
        opening = 'opening.jpg'

```

```

# Canvas
src = cv2.imread(os.path.join(path_ppr+opening), cv2.IMREAD_GRAYSCALE)
h1,w1 = OrthoYsize, OrthoXsize
canvas_1 = np.zeros((h1,w1,3),np.uint8)

# Contours 1
mode = cv2.RETR_TREE
ret_1, thresh_1 = cv2.threshold(src, 127, 255, cv2.THRESH_OTSU)
contours_1, hierarchy_1 = cv2.findContours(thresh_1, mode, cv2.CHAIN_APPROX_NONE)
approxDP_1 = []
for cnt in contours_1:
    epsilon_1 = 0.001*cv2.arcLength(cnt,True)
    approx_1 = cv2.approxPolyDP(cnt, epsilon_1, True)
    approxDP_1.append(approx_1)
    cv2.drawContours(canvas_1, [approx_1], -1, (255,255,255), -1)
#print('approxDP_1: ',len(approxDP_1))
cv2.imwrite(os.path.join(path_ppr,'cnt_approx_1.jpg'), canvas_1)

# Resize canvas
canvas_1_copy = cv2.imread(os.path.join(path_ppr,'cnt_approx_1.jpg'), cv2.IMREAD_GRAYSCALE)
height = round(OrthoYsize * scale_factor)
width = round(OrthoXsize * scale_factor)
canvas_resize = cv2.resize(canvas_1_copy, dsize=(width, height), interpolation = cv2.INTER_AREA)
h2,w2 = canvas_resize.shape[0], canvas_resize.shape[1]
canvas_resize = cv2.flip(canvas_resize,0)
#print('Resized: ',canvas_resize.shape)

canvas_2 = np.zeros((h2,w2,3),np.uint8)

# Contours 2
mode = cv2.RETR_TREE
ret_2, thresh_2 = cv2.threshold(canvas_resize, 127, 255, cv2.THRESH_OTSU)
contours_2, hierarchy_2 = cv2.findContours(thresh_2, mode, cv2.CHAIN_APPROX_NONE)
approxDP_2 = []
for cnt in contours_2:
    epsilon_2 = 0.0001*cv2.arcLength(cnt,True)
    approx_2 = cv2.approxPolyDP(cnt, epsilon_2, True)
    approxDP_2.append(approx_2)
    cv2.drawContours(canvas_2, [approx_2], -1, (0,0,255), 1)
#print('approxDP_2: ',len(approxDP_2))

cv2.imwrite(os.path.join(path_ppr,'cnt_approx_2.jpg'), canvas_2)
print(' - Contour extraction has completed')
return contours_2, h2

contours_2, h2 = contours(path_sth,path_ppr, OrthoXsize, OrthoYsize, scale_factor)

def Create_polygon(contours_2, x0, y0, h2, projection, path_pol, path_out, image_name):
    print(' - Creating polygons')
    poly = []
    for i in range(len(contours_2)):
        arc_len = cv2.arcLength(contours_2[i], True)
        #if (i > 0) and (len(contours_2[i]))>2:
        if (i > 0) and arc_len < 1500 and arc_len > 30 :
            poly.append(Polygon(np.squeeze(contours_2[i])))
    polygons = poly
    polygons = gpd.GeoDataFrame(polygons, columns = ['geometry'])
    polygons.to_file(path_pol+'con_to_pol.shp')

    Shapefile_1 = gpd.read_file(path_pol+'con_to_pol.shp')
    #print('Projection: ', projection)
    Shapefile_1 = Shapefile_1.set_crs(projection)
    print(' - Setting Coordinate system')

    # Final output
    Shapefile_2=Shapefile_1.translate(x0, y0-h2)

    Shapefile_2.head()
    output_vector = image_name[:-4]
    Shapefile_3 = Shapefile_2.geometry.simplify(tolerance=1,preserve_topology=True)
    Shapefile_3.to_file(path_out+output_vector+'.shp')

Create_polygon(contours_2, x0, y0, h2, projection, path_pol, path_out, image_name)

```

지형지물(건물) 자동인지 알고리즘을 활용한 정사영상 내 건물인지

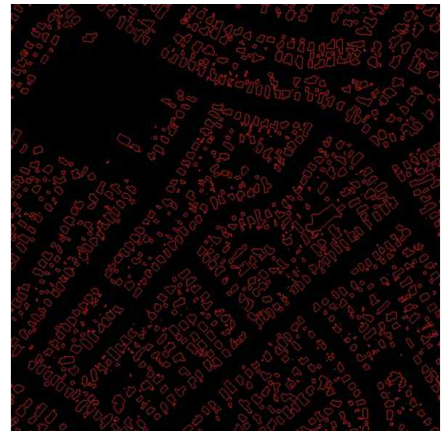
[영상공간정보내 지형지물(건물)검출]



입력 정사영상



이미지변환 적용



건물 검출 결과

[건물 벡터데이터 생성 결과]

