放置林における CO2 固定量算出に向けた計測と解析手法の構築

Development of measurement and analysis methods for calculating CO2 fixation in abandoned forests

高知工科大学大学院
工学研究科基盤工学専攻
社会システム工学コース
国土情報処理工学研究室
1265055 西岡幸亮
指導教員　高木方隆
副指導教員　赤塚慎
論文副審査　佐藤慎司

2024 年 1 月 17 日

論文要旨

近年、世界的な温暖化の影響を踏まえ、カーボンニュートラルの達成に向けた動きが加速している。日本も例外ではなく、2013 年度比で 2030 年までに 26%、2050 年までに 50%の削減目標を掲げている。そのため、現在排出されている CO_2 を森林などで循環させることが緊急の課題とされている。しかし、日本では管理されている森林での観測が多く、複雑な構造を持つ放置林では行われていない。海外では複雑な構造を持つ熱帯雨林での観測が進んでおり、放置林に対する観測の参考になるのではないかと考えた。どの先行研究でも、森林ああああ全体のバイオマス量や CO_2 固定量などでの計測としており、単木レベルでの解析は行われていなかった。そこで本研究では、放置林(自然林)を対象にした単木レベルでの CO２固定量の計測及び解析手法を確立する。

複雑な構造を持つ森林で単木レベルの解析を実現するためには、先行研究で行われている、高高度 UAV 写真観測、高高度 UAV_LiDAR 観測、地上 LiDAR に加え、本研究では低高度 UAV を導入した。これらで得られた様々なプロダクトをボクセルモデルに統合し、単木レベルでの解析が可能となる手法を開発した。

結果、低高度 UAV を林床内部で飛行させることによって、放置林においても胸高直径を求めるのに十分な情報が得られた。次に、複数機器から得られる様々なプロダクトは、ボクセルモデルを用いることにより、複数のデータを統合することが可能となり、森林の構造を詳細に表現できることが可能となった。また、ボクセルモデルを用いることで単木レベルでの CO_2 固定量の算出が可能となった。これらの事から、複雑な構造を持つ放置林に対して有効な森林計測手法とボクセルモデルを用いた解析手法を構築することができたと言える。

今回の観測対象地域においては、CO_2 固定量の変化を 2023 年と 2019 年のデータから求め、将来予測も可能となった。樹木の成長量は、北西斜面が最も樹高成長量が高く、樹冠面積も関係していると考えられた。

Abstract

In recent years, considering the global impact of climate change, there has been an accelerated movement towards achieving carbon neutrality worldwide. Japan is no exception, having set reduction targets of 26% by 2030 and 50% by 2050, compared to the levels in the fiscal year 2013. Consequently, addressing the urgent task of cycling currently emitted CO_2 through forests has become crucial. However, in Japan, observations are predominantly conducted in artificial forests, with neglected forests possessing complex structures being largely overlooked. Overseas, observations in tropical rainforests with complex structures have advanced, offering insights that could be valuable for neglected forests. Existing studies primarily focus on measurements related to overall forest biomass and CO_2 fixation. Therefore, this study aims to establish measurement and analysis methods targeting neglected forests (natural forests) for individual tree-level analysis.

To achieve individual tree-level analysis in forests with complex structures, the study introduces low-altitude UAV (Unmanned Aerial Vehicle) flights in addition to high-altitude UAV photo observations, high-altitude UAV LiDAR observations, and ground-based LiDAR, as employed. Data obtained from these sources are integrated using voxel modeling, enabling individual tree-level analysis.

As a result, deploying low-altitude UAVs for flights within the forest floor allowed the development of an effective method for neglected forests. This approach facilitates the observation of numerous neglected forests. Furthermore, by utilizing voxel modeling, various products derived from multiple devices can be integrated, allowing for a detailed representation of forest model structures. Additionally, voxel modeling enables the calculation of CO_2 fixation at the individual tree level.

In the factor analysis affecting tree growth, it was found that the northwest slope exhibited the highest tree height growth, while the south slope showed the lowest growth. The results also indicate a correlation between crown spread and tree height growth.

目次

1 序論

1.1　森林計測の現状

近年、世界的な温暖化の影響を踏まえ、2015 年に採択されたパリ協定以降、カーボンニュートラルの達成に向けた動きが加速している。カーボンニュートラルは、温室効果ガスの排出量から森林などによる固定量を差し引いて、その合計をゼロにすることを目指すものである。日本も例外ではなく、2013 年度比で 2030 年までに 26%、2050 年までに 50%の削減目標を掲げており、その達成に向けた取り組みが進行中である。しかしながら、工業化以前と比較して既に 1.1℃上昇[1]しており、これが持続すれば異常気象や環境被害が予測される。そのため、現在排出されている CO_2 を森林などで循環させることが緊急の課題とされている。日本の国土の約 67%が森林で覆われており、その CO_2 固定量は非常に大きいと見られている。2021 年時点での国土交通省による、森林の管理状況についてのアンケート調査が行われた(図 1)。

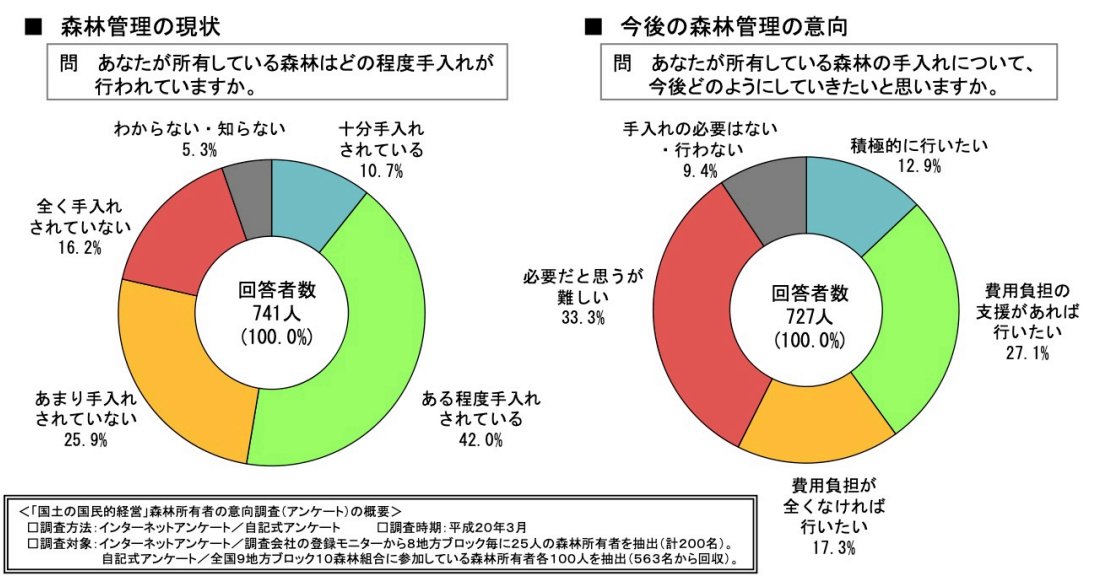


図 1 国土交通省によるアンケート調査

この調査の結果から、10.7%しか、十分に手入れされていないため、ほとんどの森林が放置林だといえる状況である。そのため放置林に対する、森林観測手法が今後必要になってくると考えられる。しかしながら、管理されている森林でも、従来の森林観測手法では、CO_2 固定量を正確に推定することが難しいとされてきた。これまでは、調査地での一部の木を対象に樹高や胸高直径を計測し、それを基に全体の CO_2 固定量を推定していた。[2]この方法では一部の木を対象としているため、全体の樹木数や密度、成長状況などを正確に把握することが難しいとされている。一方でここ数年、技術の進歩により森林観測の手法に大きな変化がもたらされている。UAV や LiDAR などの新たな技術の導入により、森林調査の正確性と効率性が飛躍的に向上している。UAV は航空写真を収集し、LiDAR は高精度な距離測定を行い、立体的な地図を生成する[3]。また、衛星データの活用も大規模な森林の変化や状態を

リアルタイムで把握する手段として注目されている[4]。これらの先端技術の活用により、森林全体の構造や成長率、炭素貯蔵量などをより正確に評価することが可能となり、これまで以上の精度で広域の CO_2 固定量を推定することが期待されている。UAV や LiDAR を活用することで、より広範囲かつ詳細なデータの収集が可能となった。UAV は空中から高解像度の画像を撮影し、その情報を基に森林の構造や樹木の密度を立体的に捉えることが可能である。同時に、LiDAR の利用により、地上からでは観測困難だった高度なデータが取得され、森林全体の立体的なマッピングが可能となった[5]。これにより、森林の健康状態や成長パターン、炭素の蓄積量などをより詳細に把握することが可能となった[2]。また、衛星データの使用により、広範囲な地域を定期的にカバーすることが可能になった。これにより、季節ごとの変化や長期的な傾向を把握することができ、植生フェノロジーを観測することが可能となった。これらの UAV を用いたデータ解析では、点群、グリッド、ボクセルで解析されている。

しかし、ボクセルを用いた解析では風を調査するものや、シミュレーション関連であり、森林解析に用いられる事例はない。そこで地上 LiDAR や低高度 UAV を用い、複雑な構造を持つ放置林に対してボクセルモデルを用いデータを統合することにより、森林の単木レベルでの CO_2 固定量を推定する計測及び解析手法構築を試みた。

近年の森林研究では、管理されている森林での観測が多く複雑な構造を持つ放置林では行われていない。海外では複雑な構造を持つ熱帯雨林での観測が進んでおり、放置林に対する観測の参考になるのではないかと考えた。管理されている森林での観測は UAV を用いて観測し、コドラート法を行っているものや [6]、ボクセルモデルを用いてシミュレーションを行なっているもの[7]があった。また、ボクセルモデルを用いて風の解析を行なっているものもあった[8]。熱帯林での解析は UAV を用いた二酸化炭素固定量の推定[9]や、全体のバイオマス量の計測を行なっているものが存在した[10]。

また、どの先行研究でも、森林全体のバイオマス量や CO_2 固定量などでの計測となっており、単木レベルでの解析に十分なデータは見受けられなかった。

本研究では今までに行われている、UAV による高高度観測、地上 LiDAR による地上観測に加えて、低高度 UAV を用いた低高度観測を行い、ボクセルモデルによる複数機器で得られたプロダクトを統合して解析することによって、そこで本研究では、放置林(自然林)を対象にした単木レベルでの CO２固定量の計測及び解析手法を確立する。

1.2　CO_2固定量の算出方法

現在行われている CO_2固定量の算出方法はいくつか存在する。まず一つは、式(a)を用いた方法となっている。[11]

幹材積×拡大係数×(1+地上部・地下部比)×容積密度×炭素含有率　　(a)

拡大係数、地上部・地下部比、容積密度は樹種ごとに異なる。これらは公開されているデータがあり、それらを用いた。この算定方法は、森林づくり活動に取り組む企業などが植栽、下刈り、除伐、間伐などの適切な施業を行った森林において、自ら算定・公表しようとする場合の標準的な計算方法を示すものである。そして、胸高直径を用いた方法として式(b)がある。[2]

$$Y=a(X+c)^b-aX^b=a\{(X+c)^b-X^b\} \qquad (b)$$

Y:1 年間の木質部幹重の成長量

X:胸高直径

a,b:相対成長式から得られる定数

c:回帰式から得られる胸高直径の年間成長量

相対成長式とは、樹高の成長と重量の成長の異なる次元の成長関係を表す式である。この式から CO2 固定量を求めることが可能となっている。

次にアメリカ南東部での算出方法は式(c)となっている。[12]

D<11 の場合:$W=0.25D^2\times H$

D>11 の場合:$W=0.15D^2\times H$

$$CO_2 = W\times 1.2\times 0.725\times 0.5\times 3.6663 \qquad (c)$$

D：胸高直径(インチ)

H：樹高(フィート)

W：木の地上部の重さ（ポンド）

この式では、樹高と胸高直径のみを用いた方法となっている。

式(b)を使うためには木を伐採し、重量を計測しなければならないため、計測を行う時点で成長を止めてしまう。また、式(c)はアメリカで用いられているため、日本で適用すべきではない。そのため,本研究では式(a)を用いることでCO_2固定量を計測する。

1.3　研究目的と構成

本研究では、先行研究では行われていなかった、複雑な構造を持つ放置林を複数機器で観測し、それらのデータをボクセルモデルで統合する。単木レベルで高精度に樹木の空間構造を計測および解析し、CO_2固定量を把握する手法を構築することを目的とした。本研究の意義は、現在多く存在すると考えられる放置林に対し、高効率で観測が行え、人的コストを削減でき、さらに高精度に単木レベルの解析が可能となることである。解析フローを図 2 に示す。

2 章では、樹高成長量の要因を解析する。樹冠部の観測には高高度 UAV 写真観測と高高度_LiDAR を組み合わせ広域観測を行う。高高度_LiDAR から作成される葉点群ボクセルモデルは、樹木 LAI(葉面積指数)計測に用いる。高高度 UAV 写真観測は高解像度の空中画像を取得し、樹木の密度や樹冠構造を立体的に捉える。これによって作成する表層ボクセルモデルは、樹冠ポリゴン作成に用いる。樹木 LAI 計測と樹頂点の抽出結果と斜面方位のデータから樹高成長量の要因解析を行う。

3 章では、CO_2固定量の推定を行う。複雑な放置林の構造をとらえるため、低高度 UAV を用い森林内部からの詳細な画像を取得し、林床部の地形や樹木構造を把握する。低高度 UAV を森林内部で飛行させるため、狭域の観測となる。そして地上 LiDAR は、地上からの測定を行い、低高度 UAV では捉えきれない森林の中層部の点群データを補う。2 つの観測機器から得られる森林内ボクセルモデルより、森林内の中層部までボクセルモデルを作成することができる。このデータを用いて樹冠部を形成する幹の抽出を行い、胸高直径計測を行う。広域観測によって得られた、樹頂点抽出の結果と合わせることによって統合ボクセルモデルを作成し、単木単位での幹材積推定を行い、CO_2固定量の算出を行う。

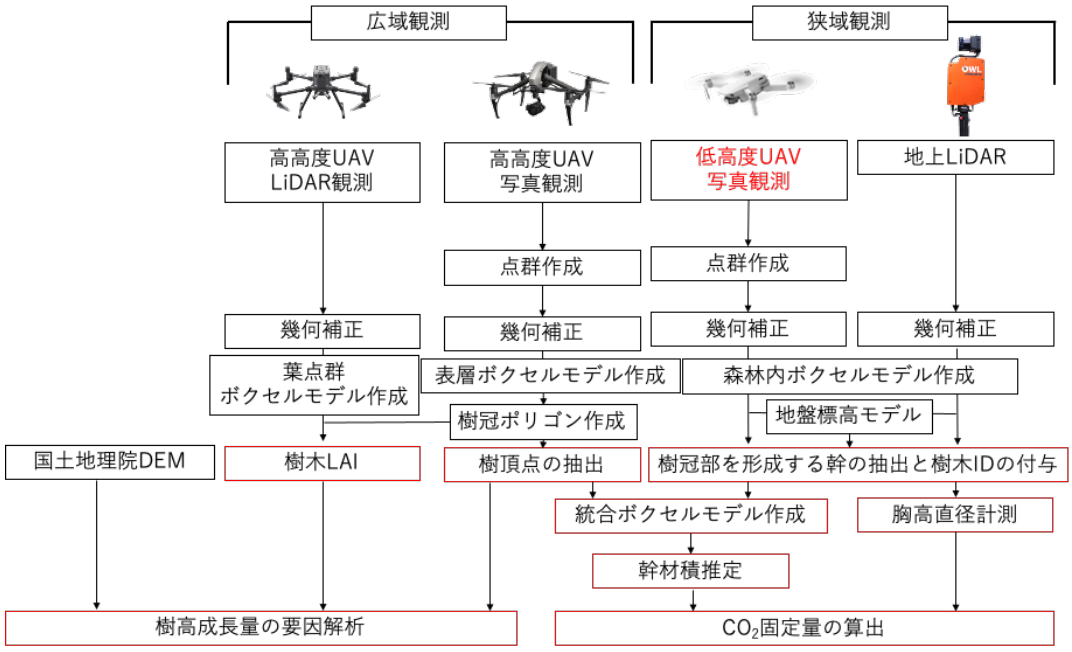


図 2 解析フロー

2　樹高成長量の要因解析

2.1　対象エリア

本研究室の森林樹冠観測は高知県香美市佐岡にある里山研究フィールドの 300m 四方のエリアで行っている。このエリアでは、2019 年から現在まで持続的な観測が行われている。標高の変化や樹木の密度、成長率など、樹冠部のパラメータを把握するため、UAV と LiDAR を活用した観測を継続している。本研究でも同様のエリアを対象とした。樹冠部の対象エリアを図 3 に示す。

図 3 樹冠対象エリア

2.2　高高度 UAV 写真観測

図 4 の DJI inspire2 での樹冠観測は搭載カメラより得た連続写真で行い、SfM での処理を行う。SfM とは Structure frome Motion の略称で複数枚の写真から 3 次元点群モデルを復元する技術である。DJI inspire2 に搭載する X5S と P1 のスペックは表 1 に示す。

図 4 DJI inspire2

表 1 DJI inspire2 に搭載する X5S と P1 の仕様

仕様	DJI X5S	DJI P1
センサーサイズ	Micro Four Thirds（MFT）	35mm
最大写真解像度	20.8メガピクセル	45メガピクセル
最大ビデオ解像度	4K/60fps	4K/30fps
ビデオビットレート	最大100 Mbps	最大200 Mbps
最大ISO	12800（写真）、6400（ビデオ）	12800（写真）、6400（ビデオ）

写真観測は DJI 社のアプリケーション DJI GO4 により自動飛行で行われる。対地高度は離陸地点で 135m で地形や樹高によって異なる、撮影画像の分解能は約 1.1cm~2.2cm となっている。観測のルートは図 5 に示す

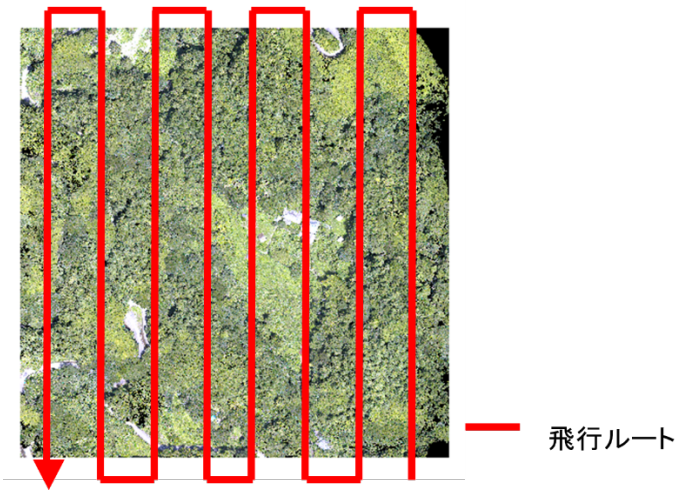


図 5　樹冠観測の飛行ルート

これらのルートで撮影された写真に SfM 処理を行う。今回使用する SfM ソフトは Agisoft 社の metashape を使用した。使用した基準点データ(GCP)を表 2 示す。作成され点群デ

ータの結果を図 6 に示す。また 2 時期の基準点周りの精度を表 3 に示す

表 2 樹冠観測での基準点(GCP)データ

Number	X	Y	Z
P1	20209.19	71715.44	126.633
P2	20270.98	71699.85	133.004
P4	20262.89	71855.74	195.884
P5	20051.35	71801.15	156.393
P7	20165.27	71558.22	131.961
P8	20248.79	71629.27	114.76

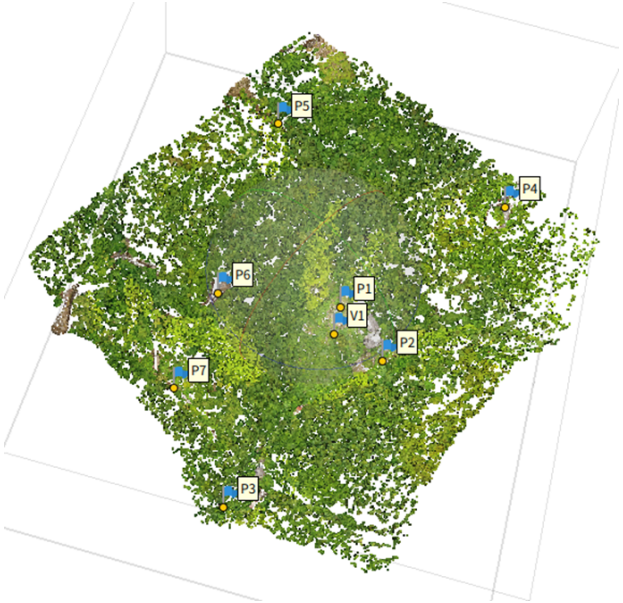


図 6　SfM で作成した樹冠点群と GCP の位置

表 3 二時期の基準点周りの精度

number	2019年誤差	2023年誤差
P1	0.061	0.014
P2	0.035	0.011
P4	0.026	0.003
P5	0.051	0.002
P7	0.012	0.023

表 4　2 時期の点群の使用

年	写真枚数	点群数	密度
2019	365	203781638	2264.24042
2023	224	193653968	2151.71076

これらのデータを用いて、樹頂点の抽出や樹冠ポリゴンの作成を行う

2.3　ボクセルモデルの作成

点群データから林床部と同じサイズのボクセルモデルを作成する。ボクセルモデルとは対象の 3 次元形状を立方体の集合で表したものである。ボクセルモデルに変換することにより、データ量を軽くし、解析をしやすくする。点群データをボクセルモデルに変換する式(c)を用いてボクセルモデルに変換する。今回 gridsize は 20 ㎝とする。これは SfM で作成した点群データの幾何精度により決定した。これらの処理を Python で作成した自作プログラムで行った。(5.1　cloud_to_Voxcel.py)

$$K = \frac{(x-xmin)}{gridsize} \quad J = \frac{(ymax-y)}{gridsize} \quad I = \frac{(z-zmin)}{gridsize} \qquad (d)$$

K J I；ボクセル番号

xmin:X 座標の最小値

ymax:Y 座標の最大値

zmin:Z 座標の最小値

x:対象点の x 座標

y:対象点の y 座標

z:対象点の z 座標

gridsize:一つあたりの格子のサイズ

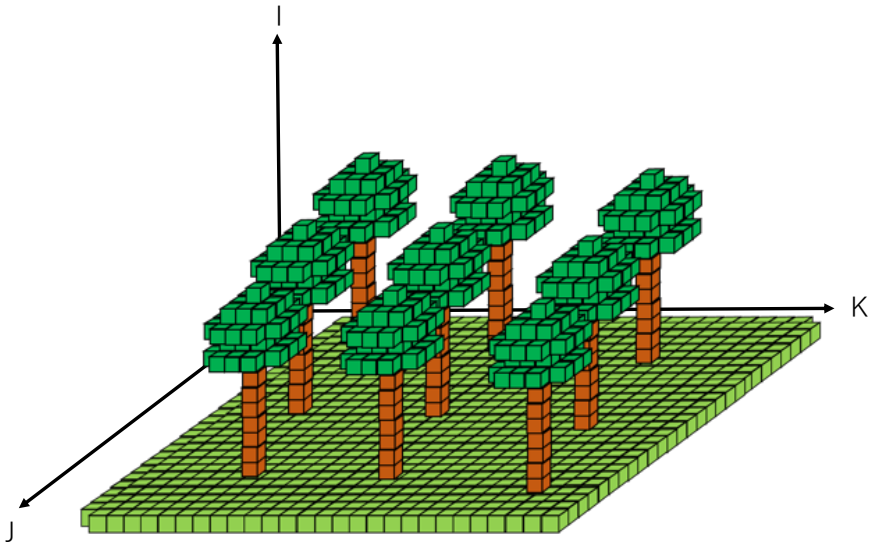


図 7　ボクセル概念図

2.4　DSM・オルソ画像の作成および樹冠ポリゴンの作成

DSM(数値表層モデル)とは、建物や樹木などの高さを含む地表面の標高を表現したデータである。オルソ画像とは、正射投影された画像であり、地図上の像の位置ずれや、傾きがなく、正しい大きさと位置に表示されているものである。作成したボクセルモデルを 2 次元に投影する。投影するときに最高点の Z 座標を格納することによって DSM を作成する。また、DSM を作成する際に格納する値を RGB 値にすることによりオルソ画像を作成することができる。このアルゴリズムを Python の自作プログラムにより作成した。(5.1 cloud_to_Voxcel.py)作成した二時期のオルソ画像を図 8 に示す。二時期の DSM 画像を図 9 に示す。

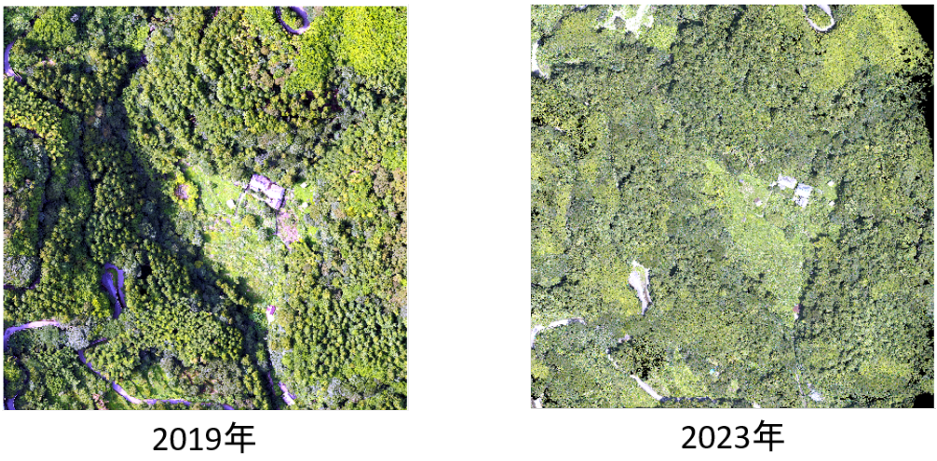


図 8 二時期のオルソ画像

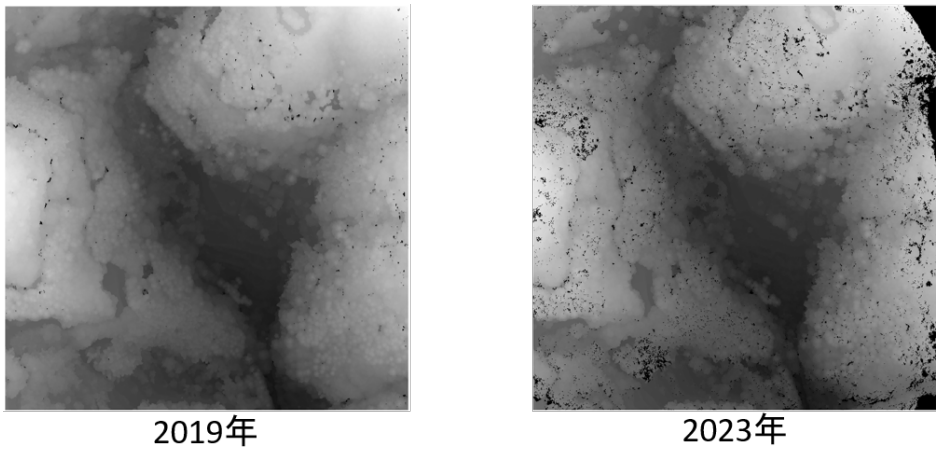


図 9 二時期の DSM 画像

表 5　オルソ画像と DSM のファイル名

	オルソ画像	DSM
2019年	2019Arutho.tif	2019DSM_TiriinDEM.tif
2023年	2023Arutho.tif	2023DSM_TiriinDEM.tif

樹木単位で成長量を解析するために樹幹ポリゴンを作成した。樹冠ポリゴンとは、樹冠のオルソ画像と DSM から目視で見える樹木の輪郭を QGGIS により手動でポリゴン化したものである。DEM（数値標高モデル）データは、地表面を等間隔の正方形に区切り、それぞれの正方形に中心点の標高値を持たせたデータである。地形図や立体地図の作成に利用される。航空レーザー測量を用いて高さを求める測量で、地表面の標高を観測している。今回作成する際に、対象エリアが斜面であるため DSM から国土地理院の DEM データを差分することで樹木の輪郭をとらえ目視でのポリゴン作成を行った。作成した結果を図 10 に示す。樹冠ポリゴンのデータ数は 2096 個となった。

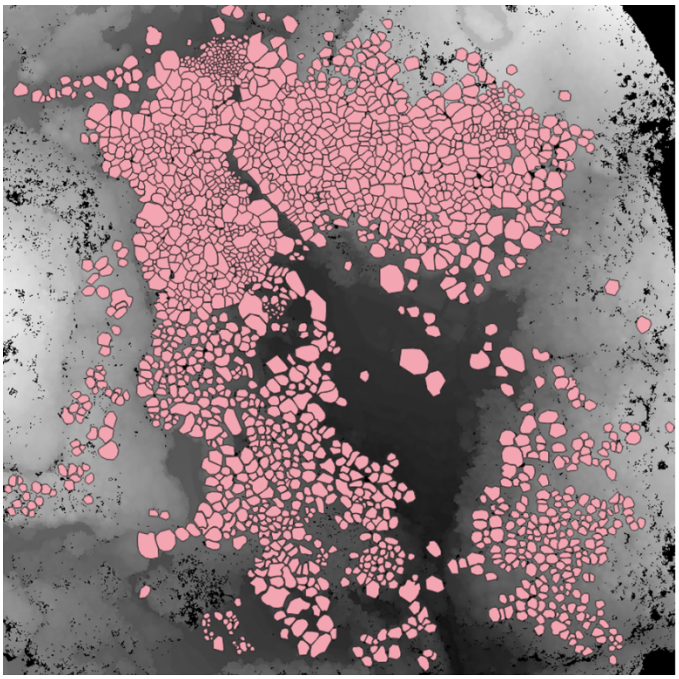


図 10　3 次元樹冠ポリゴン

2.5 樹高成長量計測手法

作成した樹冠ポリゴンを用いて樹頂点を抽出する。樹冠ポリゴン内の最高点を樹頂点とし抽出を行った。(5.3 Tree_ext.py) 抽出した 2 時期の樹頂点の Z 座標を差分することで成長量として算出した。結果を図 11 に示す。結果は 0.8～1.0ｍ成長量が最も多い結果となった。その他に関しては、マイナスの値や 2m 以上の値となっている。マイナスの値や 2m 以上の値は点群精度の問題や間伐の影響で算出されていると考えられる。

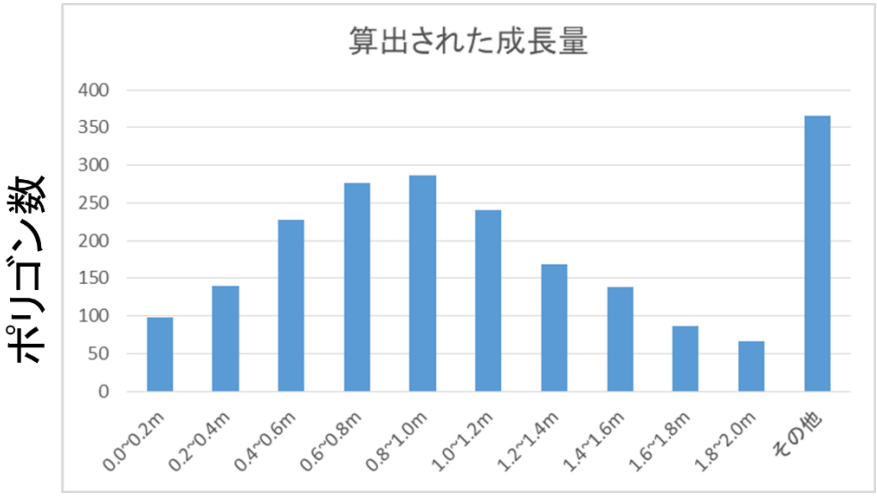


図 11 樹高成長量算出結果のヒストグラム

2.6　高高度 UAV__LiDAR 観測

LiDAR とは光を用いた観測で、パルス状に発行するレーザー照射に対する反射光より、距離などを測るものである。樹冠観測で LiDAR を用いることにより、樹冠表面だけでなく森林の内部も観測することが可能となる。LiDAR 観測では点群データが取得できる。この観測では図 12 の MATRICE300RTK を使用する。MATRICE300RTK に搭載される L1 の仕様は表 6 に示す。取得された点群データを図 13 に示す。

図 12　MATRICE300RTK

表 6　MATRICE300RTK に搭載される L1 の仕様

項目	仕様
レーザースキャナー	レーザータイプ: Time-of-Right (ToF) 、レーザークラス: Class 3、レーザー波長: 1550nm
レーザースキャナー精度	水平方向: 1cm + 1ppm、垂直方向: 1.5cm + 1ppm
カメラ	カメラセンサータイプ: CMOS、カメラセンサーサイズ: 1インチ、画像サイズ: 20メガピクセル、動画解像度: 4K @ 30fps
GNSS	受信機: GPS + GLONASS + BeiDou + Galileo
レーザーレンジファインダー	レンジ: 10cm - 100m、精度: 1cm + 1ppm
IMU	レーザースキャナーと一体化された6軸IMU

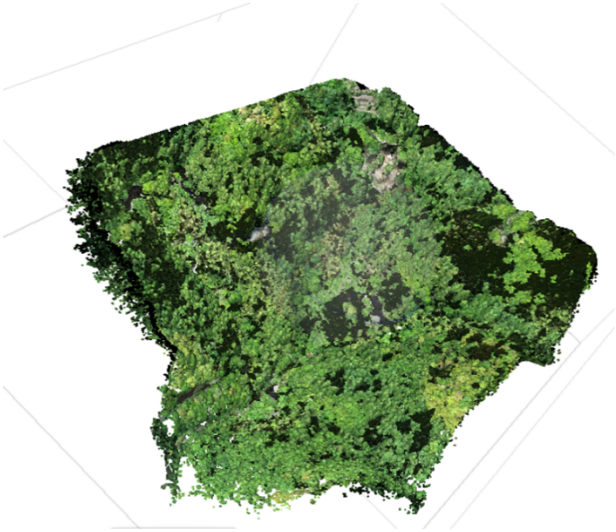


図 13　取得された樹冠点群データ

表 7　LiDAR 点群の仕様

日付	点群数	密度（点/m2）	ファイル
2023/7/29	153019002	1700.211133	20230729.las

2.7　LAI 計測手法

LAI を計測する。LAI（葉面積指数）は、葉の面積を示す指標であり、葉は光合成などの重要な生態学的機能を担う部位である。このため、LAI と成長量との間には密接な関係があると考えられる。LAI の計測には、従来の手法としては森林内の光から間接的に求める LAI-2000 というものを使う。しかし、今回は、直接計測した、葉点群を用いた方法を提案する。葉点群のみを抽出するための式(e)を用いて、葉点群のみの抽出を行った。(5.1 cloud_to_Voxcel.py)次に各樹冠ポリゴンごとに点群を区分し、ポリゴン内の点群数を求めそれ LAI とする点群数 LAI を算出した。LiDAR での計測は UAV スピードが一定なら点群が均一であるため、点群数での LAI を求めた。今回は 2023 年の葉点群で LAI 計測を行う。結果を図 14 に示す。

$$Leaf=R<G \qquad (e)$$

R:点群の赤バンドの強度
G:点群の緑バンドの強度

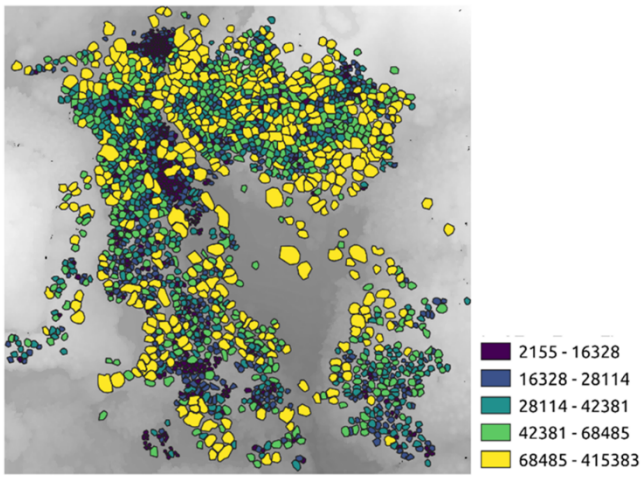

2155 - 16328
16328 - 28114
28114 - 42381
42381 - 68485
68485 - 415383

図 14 算出した LAI

2.8　樹高成長量に影響を与える要因解析

QGIS を用いて DEM から斜面方位を算出した。DEM データを国土地理院から取得し、QGIS で DEM から斜面方位を算出した(図 15　斜面方位)。対象エリアは 2.1 と同じである。

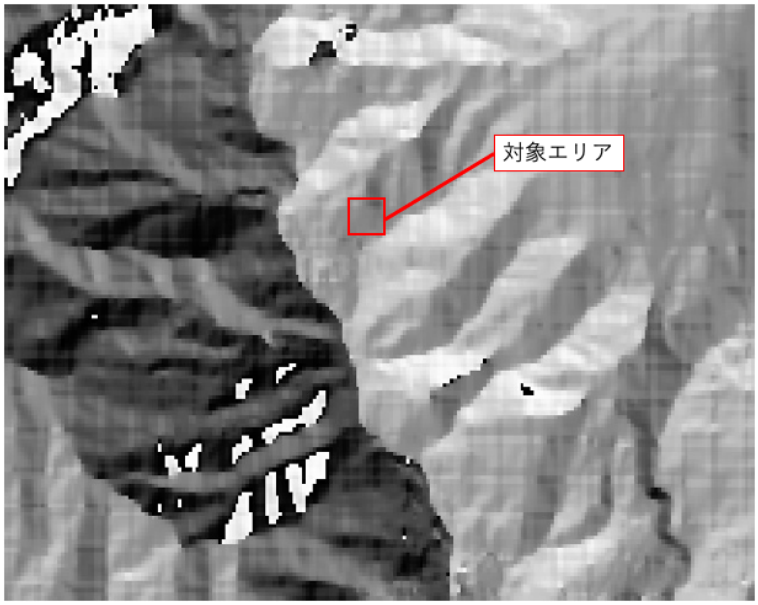
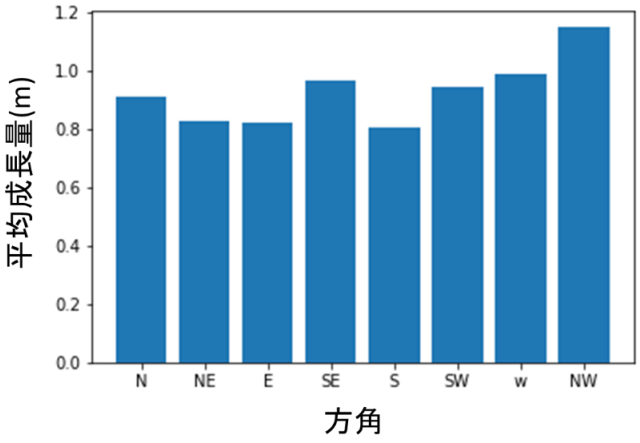


図 15　斜面方位

各セルごとに、その地点から最も急傾斜な方向を示す。得られた斜面方位データは、8 つの方位（北、北東、東、南東、南、南西、西、北西）に分類した。斜面方位と成長量の情報を組み合わせて、各ポリゴンごとに斜面の傾斜が成長量に与える影響を 8 方位ごとに詳細に解析した。成長量がマイナスの値や、2m を超えていたものに関しては除外している。結果を図 16 に示す。北西斜面が最も成長量が高く、南斜面が成長量が低い結果となった。



方位	データ数
北(S)	64
北東(NE)	307
東(E)	321
南東(SE)	72
南(S)	259
南西(SW)	430
西(W)	191
北西(NW)	37
合計	1681

図 16 各方位ごとの樹高成長量のヒストグラム

LAI の変動が成長量に与える影響を把握するために、LAI と成長量の散布図より比較を行った。結果を図 17 に示す。

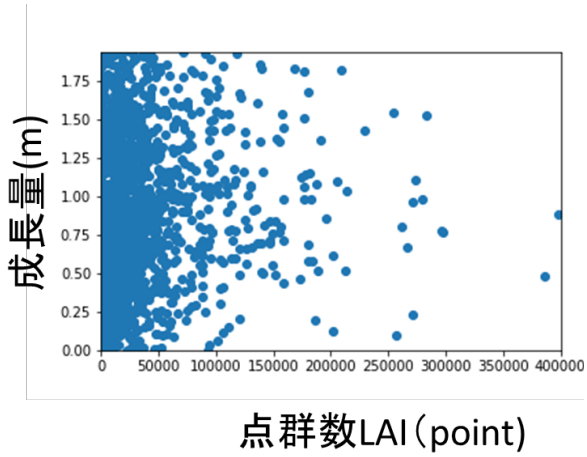


図 17　LAI と樹高成長量の比較

次に樹木密度と成長量の比較を行う。樹木密度は今回、抽出した対象の樹頂点から半径 5m 以内の樹頂点の数を樹木密度とした。成長量は、樹木密度ごとの平均成長量として比較を行った。結果を図 18 に示す。結果は、19 本での成長量が最も高い結果となった。

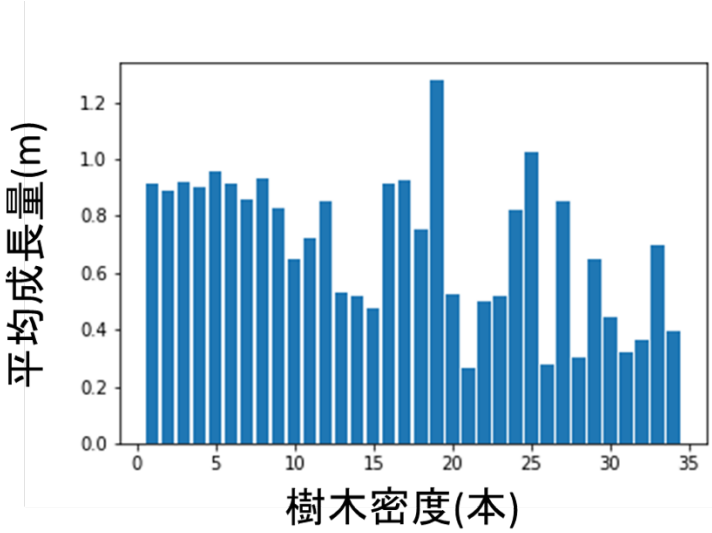


図 18　樹木密度と樹高成長量の比較

次に樹冠ポリゴンの面積と成長量を比較する。樹冠ポリゴンの面積は単木の葉の広がりを示す指標であると考え、ポリゴン面積が大きければ、葉は大きく広がっており、より光合成していると考えられる。そのため、ポリゴン面積と成長量を比較する。結果を図 19 に示す。結果はポリゴン面積と成長量に関係性は見られなかった。

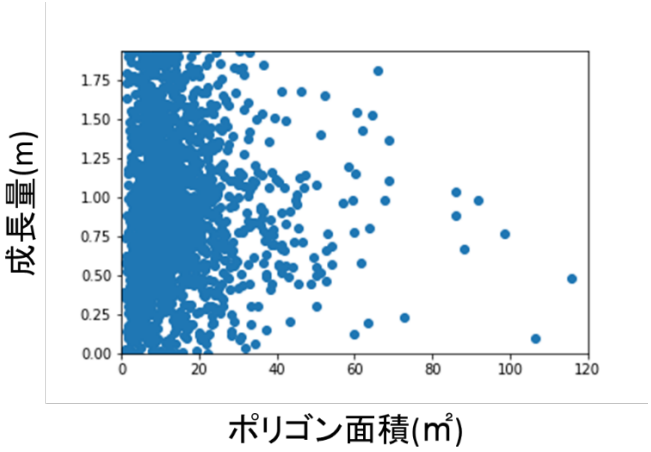


図 19　ポリゴン面積と樹高成長量の比較

次に、樹冠ポリゴンを用いて 2019 年 2023 年の DSM を差分し、樹冠の面的な広がりと、樹頂点の成長量の比較を行う。面的な広がりの結果は、ポリゴン面積が大きく影響していることを考え、単位面積あたりに換算することにより広がりと、成長量を比較した。結果を図-19 に示す。結果は、単位位面積当たりの成長量と樹頂点の成長量には関係があることが分かった。

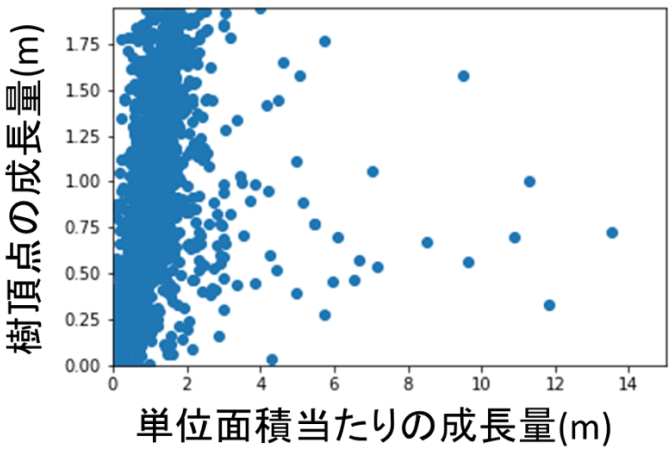


図 20　樹冠の広がりと樹高成長量の比較

3　CO_2固定量の推定

3.1　対象エリア

今回の対象地域は高知県香美市佐岡にある研究フィールドの金嶺神社周辺の 50m 四方のエリアを対象とした。このエリアには間伐したエリアと、間伐してないエリアがあるため変化が見られると考えてこのエリアを選定した。選定エリアを図 21 に示す.

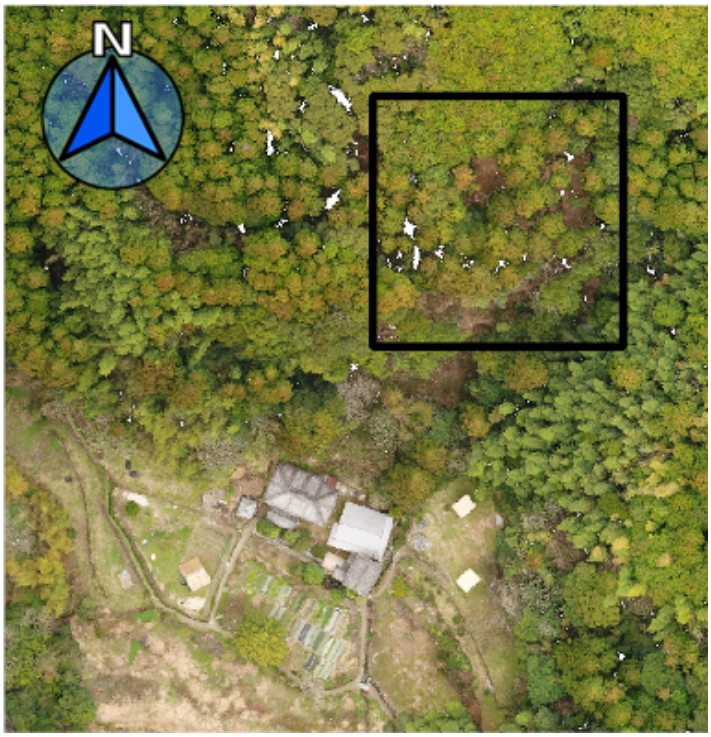


図 21　林床観測エリア

3.2　低高度 UAV 写真観測

図 22 の DJI　mini2 での観測は写真で行い、樹冠部と同様 SfM での処理を行う。DJI mini2 のスペックを表 8 DJI　mini2 仕様 表 8 に示す。

図 22 DJI　mini2

表 8 DJI　mini2 仕様

機能/仕様	詳細
カメラ	1/2.3インチ CMOS センサー、12メガピクセル
レンズ	FOV（視野）83°、35mm フォーカス距離相当、f/2.8
最大写真解像度	4000×3000 ピクセル
最大ビデオ解像度	4K @ 30fps
シャッタースピード	1/8000秒 ~ 4秒
重量	約249 g
ビデオフォーマット	MP4（H.264/MPEG-4 AVC）
写真フォーマット	JPEG

樹冠部と異なり、手動での撮影となっている。飛行ルートを図 23 に示す。観測ルートは毎回変わるが、SfM で点群を作成できる程度の写真枚数を撮影する。2021 年 8 月 11 日の観測では、1524 枚の写真を撮影した。

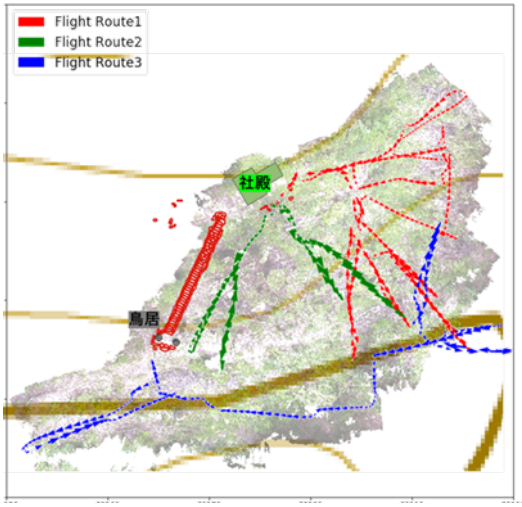


図 23　林床観測の飛行ルート

撮影された写真に SfM 処理をすることで点群データを作成する。作成された点群データを図 24 に示す。作成に使用した GCP と点群精度を表 9 に示す

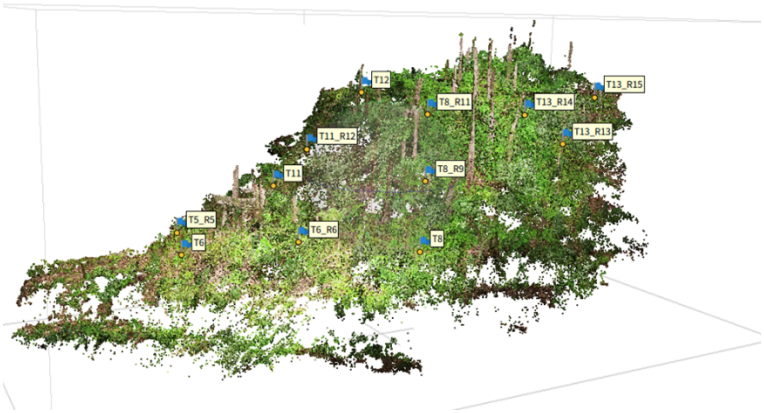


図 24 林床部の点群データ作成

表 9 林床部の GCP と点群精度

number	X	Y	Z	誤差(m)
T6	20263.7	71760.29	148.77	0.029
T8	20284.7	71764.72	149.40	0.014
T11	20269.82	71768.52	153.96	0.034
T12	20276.14	71779.21	162.30	0.011
T5_R5	20262.63	71761.77	150.40	0.010
T6_R6	20274.29	71761.04	150.48	0.026
T8_R9	20284.24	71770.4	154.75	0.081
T11_R12	20272.06	71772.39	157.06	0.030
T13_R14	20292.94	71784.26	160.13	0.031
T13_R15	20300.46	71792.33	161.57	0.011

表 10 林床点群の仕様

日付	写真枚数(枚)	点群数(point)	点群密度(point/m2
2020/7/11	669	74211242	29684.4968
2020/8/21	521	51995143	20798.0572
2020/9/9	638	63566530	25426.612
2020/10/16	580	55855456	22342.1824
2020/11/13	563	55162704	22065.0816
2020/11/25	451	47523198	19009.2792
2020/12/7	945	229199247	91679.6988
2020/12/14	934	220191305	88076.522
2021/3/4	893	160823089	64329.2356
2021/5/12	974	246619261	98647.7044
2021/6/11	594	144360406	57744.1624
2021/8/11	1524	381300896	152520.3584
2021/9/7	1128	249317831	99727.1324

3.3　地上 LiDAR 観測

図 25 の地上 LiDAR　OWL は赤外線レーザーを用いた軽量・コンパクトな一脚型の森林計測装置で、複数点での観測データをもとに立体合成を行う。観測日は 2022 年 8 月 5 日で観測点は 12 点となっている。OWL のスペックを表 11 に示す。観測点を図 26 に示す。これらのデータも点群データとして取得することができる。取得された点群データを図 27 に示す。

図 25 OWL

表 11　OWL 仕様

計測精度	平均樹高：誤差 1m 程度
レーザースキャナ	スキャン時間 45 秒設定時 86,400 点/sec
	スキャン時間 22.5 秒設定時 86,400 点/sec
サイズ重量	3.20kg

図 26 OWL 観測点

図 27 地上 LiDAR で作成された点群データ

3.4　林床のボクセルモデルの作成

DJI mini2 と OWL から取得した点群データからボクセルモデルを作成する。異なる観測機器での点群の重ね合わせには、2023 年度卒業論文の観測機器の異なる点群データの重ね合わせ手法[13]を用いて、テンプレートマッチングとアフィン変換により点群を重ね合わせた。重ね合わせた点群の精度を表 12 に示す。今回 gridsize は 10 ㎝とする。これらの処理を Python で作成した自作プログラムで行った。(5.1　cloud_to_Voxcel.py)

表 12　重ね合わせた点群の精度

処理回数	X残差(px)			Y残差(px)			Z残差(m)		
	平均	最大	最小	平均	最大	最小	平均	最大	最小
1	1.1	2	0.0	1.3	2	0.0	0.034	0.072	0.010
2	0.9	4	0.0	0.3	1	0.0	0.010	0.031	0.000
3	0.6	3	0.0	0.6	4	0.0	0.014	0.090	0.000
4	0.3	1	0.0	0.6	4	0.0	0.019	0.130	0.000
5	0.4	2	0.0	0.1	1	0.0	0.002	0.006	0.000
6	0.2	1	0.0	0.1	1	0.0	0.002	0.007	0.000

表 13　OWL 点群の仕様

点群数	30236676point

表 14　ボクセルの仕様

バンド	属性
1	x座標
2	y座標
3	z座標
4	赤バンド
5	青バンド
6	緑バンド

3.5　点群データから DEM データの作成

作成したボクセルモデルを 2 次元平面に投影することにより、DEM を作成する。2 次元投影するときに格納する値は、対象ピクセルに該当する点の Z 座標が最も低い点を使うことにより DEM データを作成した。グリッドサイズは 10cm となっている。点群が存在しない場合、欠損値となるが、逆距離荷重法による内挿処理を行う事により欠損値を内挿した。これらの処理を Python で作成した自作プログラムで行った。(5.2　IDW.py)作成した結果を図 28 に示す。

図 28 林床部の 10cmDEM データ

3.6 単木の ID 付与

作成した林床ボクセルモデルと DEM データを用いて林床部の単木抽出を行う。単木抽出のアルゴリズムでは、DEM から垂直方向の連なりが 3m 以上の物を木と定義し、抽出を行った。抽出を終えた単木に ID の付与を行った。この際に、単木の最下点と最高点も取得する。これらの処理を Python で作成した自作プログラムで行った。(5.3 Tree_ext.py)単木抽出の結果を図 29 に示す。結果 149 本の木に ID を付与することができた。

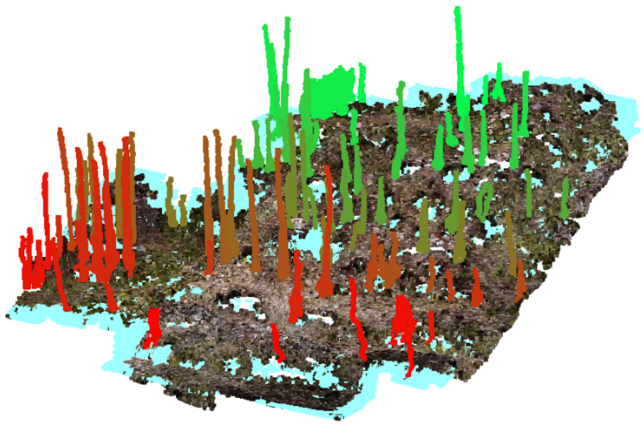


図 29　単木の抽出結果

3.7　胸高直径の推定

抽出した単木のデータを用いて胸高直径の抽出を行った。対象の単木の DEM 座標から 1.2m 地点の断面図を切り抜きそのエリアの 1cm ボクセルを作成した。1cm ボクセルを作成する際に、幹の点群データが密でないため、2 年分の DJI　mini2 の写真観測による点群データを重ね合わせ推定を行った。これらの処理を Python で作成した自作プログラムで行った。(5.4 diameter.py)切り抜いた画像の結果を図 30、胸高直径の結果を図 31 に示す。結果は 0.6m~0.8m が最も大きい結果となった。

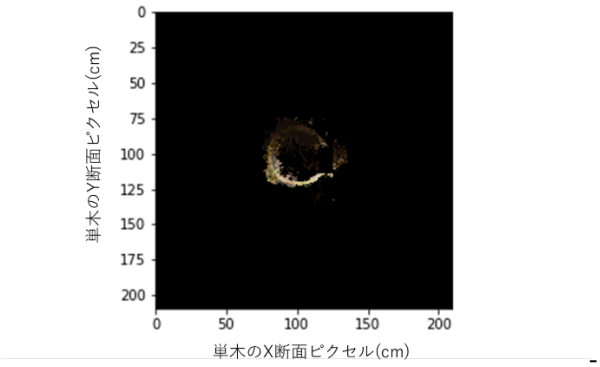


図 30　切り抜いた断面の画像

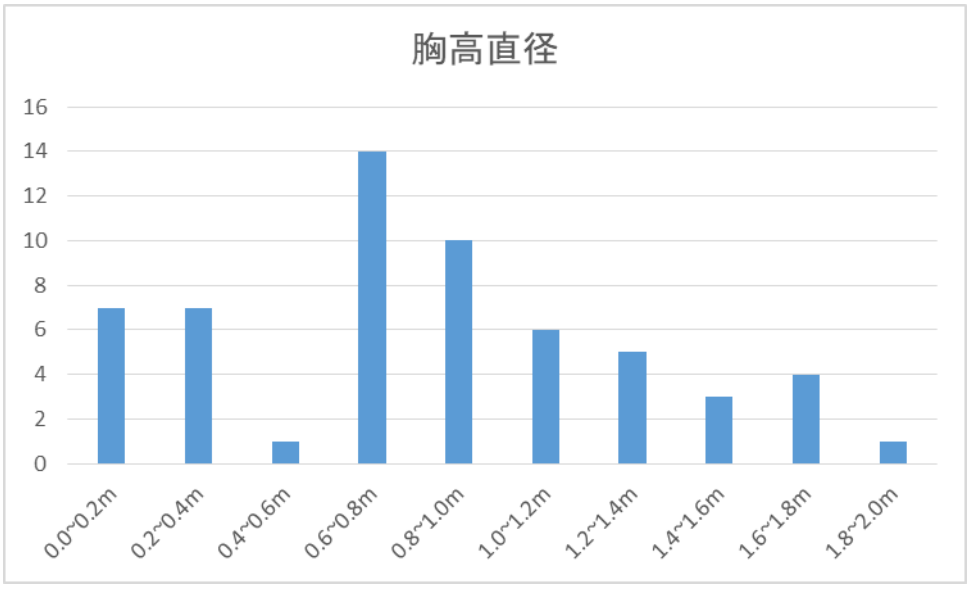


図 31　胸高直径グラフ

3.8　単木と樹冠ポリゴンの結合

樹冠ポリゴンを用いて、ポリゴン内にある幹を抽出し、その中で最高点が最も高いものが樹冠を形成している木として判定し、最下点と樹頂点を結合した(5.4 diameter.py)。結合結果を図 32 に示す。樹高計測結果を図 33 に示す。樹冠ポリゴンが作成されているが、ラインデータが存在しないものは、林床部での幹抽出が行えていないものとなっている。樹冠を形成している木は 58 本となった。(5.5 CO2 固定量属性データ)

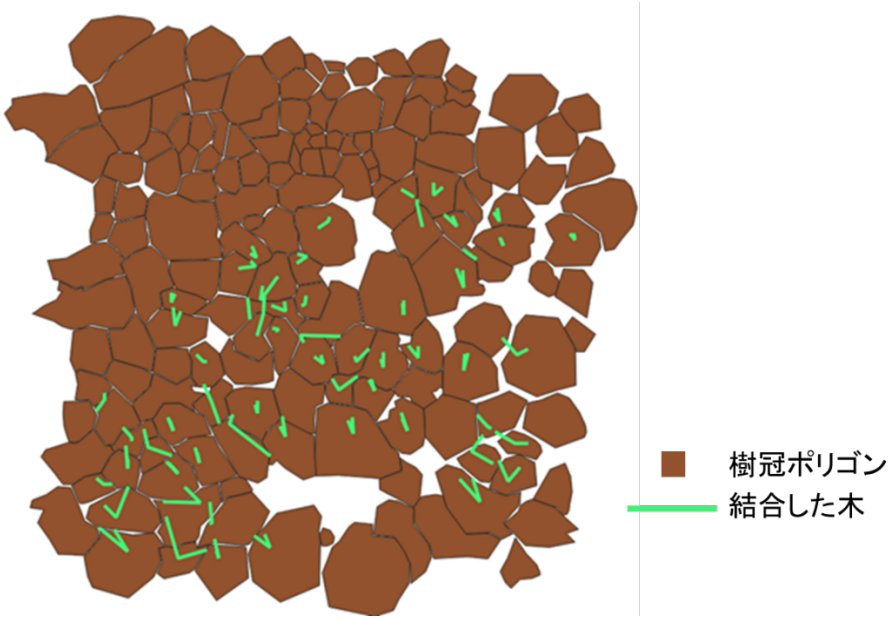


図 32 最下点とポリゴンの結合

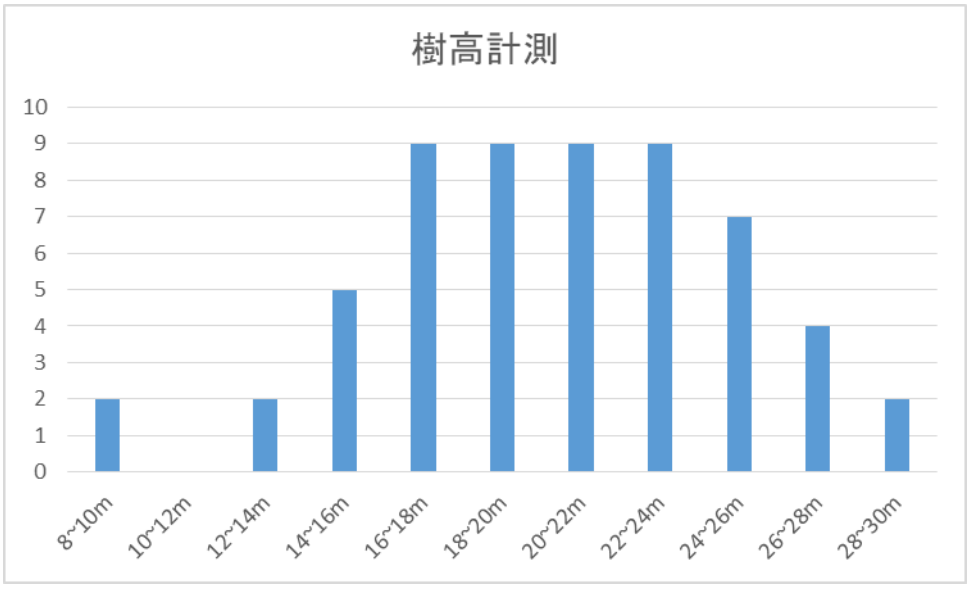


図 33　樹高計測結果

3.9　CO_2 固定量の推定

今回 CO_2固定量を算出するのはスギのみに限定する。スギ以外の樹木には、活用法が少ないため、スギに限定することとした。幹材積を推定するのに木のモデルとして図 34 のように円錐モデルを使用する。胸高直径を底面の直径とし、樹高を高さとして、求めた体積を幹材積として ID ごとに付与する。結果を図 35 に示す。結果として 2~4 ㎥の木が最も多いことが分かった。

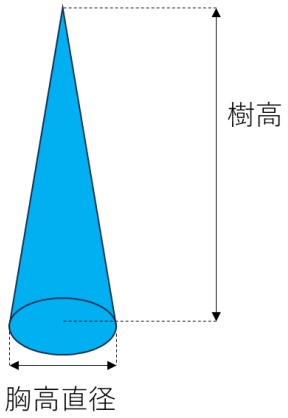


図 34　円錐モデル

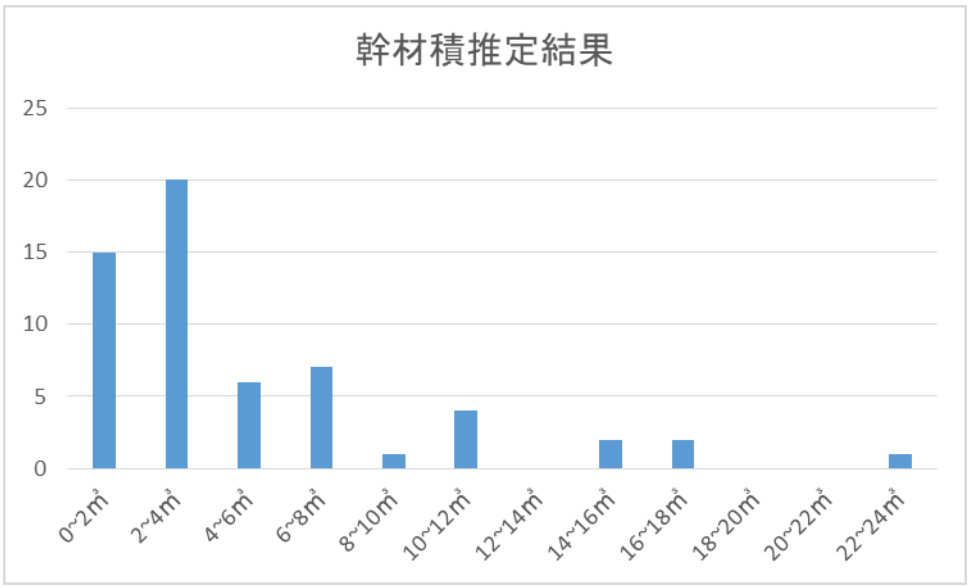


図 35　幹材積推定結果

次に式(a)を用いて 2019 年と 2023 年の CO2 固定量の算出を行う。式(a)で用いる係数などは表-9 に示す[14]。結果は図 36　2 時期の CO_2固定量に示す。

表 9 CO_2固定量算出のパラメータ

樹種	拡大係数	地上部・地下部比	容積密度
スギ	1.23	0.25	0.314

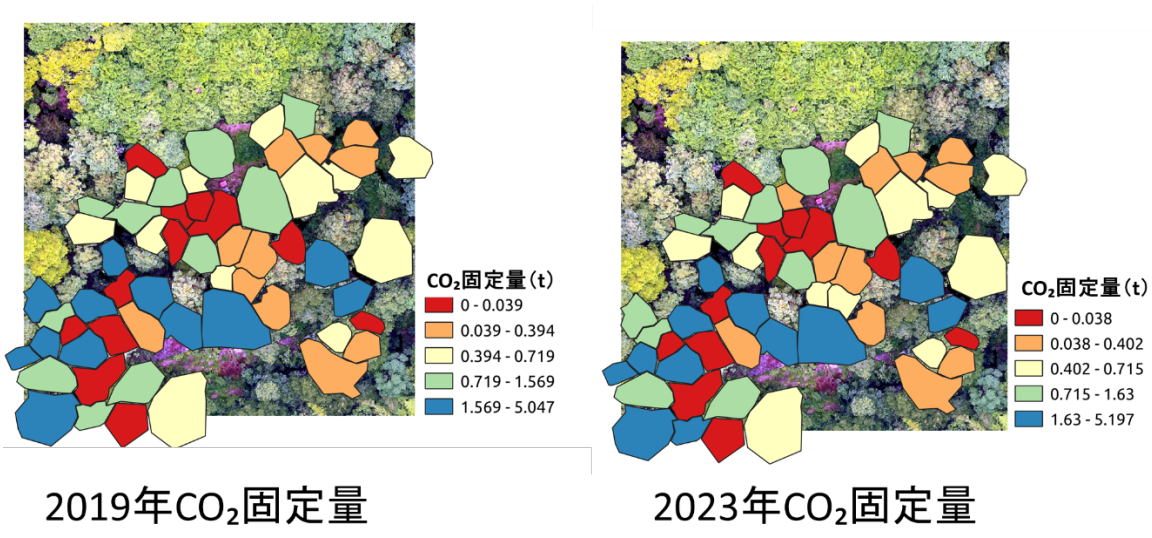


図 36　2 時期の CO_2固定量

2 時期での CO_2固定量の変化は**図 37** のようになっている

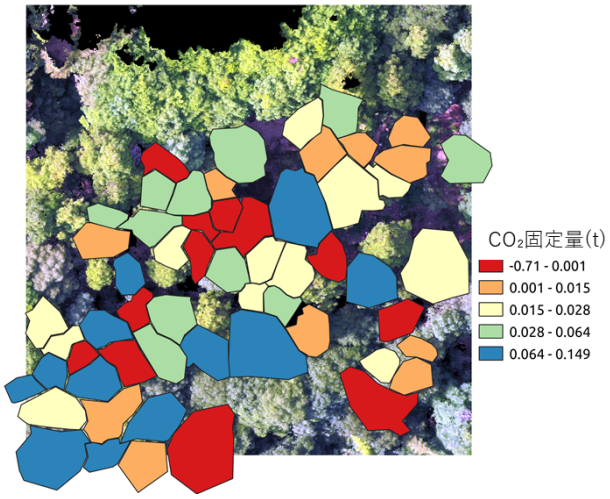


図 37　2 時期の CO_2固定量の変化

4　結論

4.1 成果

今回の研究では、低高度 UAV を林床内で飛行させることによって放置林に対する効果的な手法を構築することができた。これによって国内に数多く存在する放置林を観測することが可能となった。

次に、複数機器から得られる様々なプロダクトは、ボクセルモデルを用いることにより、林床と樹冠のデータを統合することが可能となり、森林モデルの構造を詳細に表現できることが可能となった。また、ボクセルモデルを用いることで容易に幹材積の推定が行え、CO_2固定量の算出が可能となった。

これらの事から、放置林に対して先行研究などでは行われていなかったボクセルモデルを用いた森林計測手法及び解析手法を構築することができた。

成長量に影響を与える要因解析では北西斜面が最も樹高成長量が高く、南斜面の樹高成長量が最も低い結果となり、樹冠との広がりと樹高成長量にも関係があることが分かった。

4.2 課題

胸高直径計測の結果に以上に大きい値が存在するこれは、低高度 UAV を林床内で飛行させることによって放置林に対する効果的な手法を構築することができたが、森林によっては飛行ルートが制限され点群密度にばらつきが生じ、点群精度の影響で発生すること考えられる。

次に、LAI について、点群を用いることで、直接求めることが可能となったが、検証が不十分である。大規模な範囲での LAI 検証方法を検討できていない。

引用文献

[1]カーボンニュートラルとは-脱炭素ポータル（環境省）
https://ondankataisaku.env.go.jp/carbon_neutral/road-to-carbon-neutral/
[2] 林野庁　https://www.rinya.maff.go.jp/
[3]Mathworks https://jp.mathworks.com/discovery/lidar.html
[4]宙畑　https://sorabatake.jp/31311/
[5]LiDARDEM を用いた表層崩壊のアセスメントに 適する勾配と凹凸度の計算範囲の推定 岩橋純了・神谷 泉・山岸宏光
[6]UAV を活用した広域コドラート調査の試み,赤沼隼一， 淺野保夫， 戸田満， 本多泰章, 野田敦夫， 板野友和， 山﨑渓
砂防学会誌，2021
[7] ボクセルモデルを用いた森林における太陽光反射シミュレーション，藤原 匠， 高木方隆，2019 年 58 巻 4 号 p. 184-195
[8]Voxcel sapace analaysis of terrestrial laser scans in forests for wind field modeling, A. Bienert, Ronald Queck,2010 年
[9] Estimating Aboveground Carbon Stock at the Scale of Individual Trees in Subtropical Forests Using UAV LiDAR and Hyperspectral Dat, Haiming Qin,2021 年
[10] Tree height in tropical forest as measured by different ground, proximal, and remote sensing instruments, and impacts on above ground biomass estimates, Gaia Vaglio Laurin,2019
[11] 森林総合研究所　https://www.ffpri.affrc.go.jp/
[12]EcoMatcher https://www.ecomatcher.com/how-to-calculate-co2-sequestration/
[13]暮沼,観測機器の異なる点群データの重ね合わせ手法,2023 年,高知工科大学　国土情報処理工学研究室　卒論
[14] 森林総合研究所　https://www.ffpri.affrc.go.jp/

参考文献

1)森林総合研究所　https://www.ffpri.affrc.go.jp/
2)林野庁　https://www.rinya.maff.go.jp/
3)Inspire 2-製品情報- dji https://www.dji.com/jp/inspire-2/info
4)MATRICE300RTK -製品情報 https://enterprise.dji.com/jp/matrice-300/specs
5)DJI Mini2 -仕様- dji https://www.dji.com/jp/mini-2/specs
6) 森林３次元計測システム OWL 仕様
https://www.owl-sys.com/wp-content/uploads/2023/12/SH-AME-OL200-02.pdf
7)航空機 LiDAR による単木パラメータの抽出,2004,高橋 與明
8) 田村 太壱ら， 低高度 UAV による空撮と SfM を用いた樹高計測， 2015 年,日本緑化工学会誌,p163-168

9)林 真智,衛星リモートセンシングを利用した樹高および森林バイオマスの広域評価に関する研究,2015 年, 北海道大学. 博士(農学) 乙第 6971 号
11) 野口 麻穂子, 酒井 敦, 奥田 史郎, 稲垣 善之, 深田 英久, 四国地方のヒノキ人工林における間伐後 6 年間の林床植生変化,2009 年,51 巻 2 号 p. 127-136
12) 稲富 佳洋ら,阿寒国立公園におけるエゾシカ生息密度の低下に伴う林床植生の変化,2012 年, 17 巻 2 号 p. 185-197
13) 飯田真一ら,筑波大学陸域環境研究センターに隣接するアカマツ林の胸高直径と立木密度の変化について,2001 年, 筑波大学陸域環境研究センター報告 No.2 1~6
14) 石井 孝, 梨本 真, 下垣 久,衛星データによる葉面積指数 LAI の推定,1999 年,12 巻 3 号 p. 210-220
15) Current forest carbon fixation fuels stream CO2 emissions,2019 年, A. Campeau ら

16)An estimation of CO2 fixation capacity in mangrove forest using two methods of CO2 gas exchange and growth curve analysis,2007 年,Yosuke Okimoto ら
17)Estimating Aboveground Carbon Stock at the Scale of Individual Trees in Subtropical Forests Using UAV LiDAR and Hyperspectral Data,2021 年, Dr. Haiming Qin ら
19)Using lightweight unmanned aerial vehicles to monitor tropical forest recovery2015 年, Rakan A. Zahawi ら
20)Tree height in tropical forest as measured by different ground, proximal, and remote sensing instruments, and impacts on above ground biomass estimatesGaia Vaglio Laurin,2019 年 Gaia Vaglio Laurin ら
21)Mapping forest canopy height globally with spaceborne lidar,2011 年,Marc Simard ら
22)スギおよびヒノキ人工林におけるドローン調査を想定した 立木サイズ推定手法の検討,2022 年, 江口則和ら
23)UAV を活用した広域コドラート調査の試み, 砂防学会誌 Vol73 No6 p39－42 2021 年, 赤沼隼一ら

5 付録

5.1 cloud_to_Voxcel.py

ライブラリは以下の図 38 に示す。

```
import numpy as np
from tqdm import tqdm
import os
import glob
import laspy
from concurrent.futures import ProcessPoolExecutor
import sys
name_List=["/media/owner/nisioka/L1-RandomPointcloud/20230729-1222-L1/lidars/terra_las/cloud99c2acb52b0dacf3.las"]
#name_List=["/media/owner/nisioka/L1-RandomPointcloud/20230729-1251-L1/lidars/terra_las/cloud53991d6b18bcd0ed.las"]
#name_List=["/media/takagi/nisioka1/L1-RandomPointcloud/20230729-1222-L1/lidars/terra_las/cloud99c2acb52b0dacf3.las"]
name=name_list[0]
```

図 38　cloud_to_Voxcel のライブラリ一覧

次にメイン処理の関数を示す。以下の図 39 に示す。

```
def main (cpu):
    cpu=int(cpu)
    z_correction=37.3376
    #z_correction=0
    Range=np.arange(0,len(las),1)
    Range=np.array_split(Range,cpu_num)
    #print(Range)
    Range=Range[cpu]
    arr=np.zeros((Jsize,Ksize))
    print(Range)
    count=0
    for num in tqdm(Range):
        x=las.X[num]
        y=las.Y[num]
        z=las.Z[num]
        r = las.red[num]  / 256
        b = las.blue[num]  / 256
        g = las.green[num]  / 256

        x= (x * x_scale) + x_offset
        y= (y * y_scale) + y_offset
        z= (z * z_scale) + z_offset-z_correction

        I=int(round((z-zmin)/gridsize,0))
        J=int(round((ymax-y)/gridsize,0))
        K=int(round((x-xmin)/gridsize,0))
        #print(x,y,z,r,g,b)

        if 0<=J<Jsize and 0<=K<Ksize:
            if r <g:
                arr[J][K]=arr[J][K]+1


    #print("ギガ数",sys.getsizeof(arr2)/(10**9))
    #print(cpu)
    #print("--------------------------------")
    return arr
```

図 39　cloud_to_Voxcel メイン処理

この処理では 1 行ずつ点群データを読み込みボクセルに変換している。グリッドサイズと、各座標の最大値、最小値を入力することでボクセルを生成する。グリッドサイズがあまりに細かい場合や、対象範囲が広すぎる場合は生成することができない。

5.2　IDW.py

IDW のライブラリ一覧を図 40 に示す。

```
import numpy as np
import pandas as pd
from tqdm import tqdm
import glob
import os
import cv2
import matplotlib.pyplot as plt
```

図 40 IDW のライブラリ一覧

次に、IDW のメイン処理関数を図 41 に示す。

```
def IDW (Date,savedirpath,gridsize):
    Size_voxcel=np.load("Voxcel_size.npy")
    Isize=Size_voxcel[0]
    Jsize=Size_voxcel[1]
    Ksize=Size_voxcel[2]
    Range=np.load("Range.npy")
    xmin,xmax=Range[0][0],Range[0][1]
    ymin,ymax=Range[1][0],Range[1][1]
    zmin,zmax=Range[2][0],Range[2][1]
    serch_range=3#m
    voxcel_rgb=np.load(savedirpath+"/"+Date+"/voxel/voxcelrgb.npy")
    row,col=int(round((serch_range/gridsize),0))+1,int(round((serch_range/gridsize),0))+1
    w_arr=np.zeros((row,col))
    print(np.shape(w_arr))
    c_row,c_col=int(round(row/2))-1,int(round(col/2))-1
    c_i=c_row*gridsize
    c_j=c_col*gridsize
    for i in range(0,row):
        I=i*gridsize
        for j in range(0,col):
            if c_row == i and c_col == j:
                pass
            else:
                J=j*gridsize
                i_d_square=(I-c_i)**2
                j_d_square=(J-c_j)**2
                d_square=(i_d_square+j_d_square)**0.5
                w=1/d_square
                w_arr[i][j]=w
    #print(w_arr)
    #plt.imshow(w_arr)

    serch_range=int(round((serch_range/gridsize)/2,0))
    dem=np.load(savedirpath+"/"+Date+"/DEM/DEM_Filter_znum.npy")
    newdem=np.zeros((Jsize,Isize))
    for j in tqdm(range(serch_range,Jsize-serch_range)):
        for k in range(serch_range,Ksize-serch_range):
            if dem[j][k] == 0:
                extract=dem[j-serch_range:j+serch_range+1,k-serch_range:k+serch_range+1]
                Row,Col=np.shape(extract)
                #print(np.shape(extract))
                wz_sum=0
                w_sum=0
                for row in range(0,Row):
                    for col in range(0,Col):
                        if extract[row][col] != 0:
                            if w_arr[row][col] != 0:
                                z=extract[row][col]
                                wz_sum=wz_sum+z*w_arr[row][col]
                                w_sum=w_sum+w_arr[row][col]

                if w_sum != 0:
                    newdem[j][k]=wz_sum/w_sum
            else:
                newdem[j][k] = dem[j][k]
    for j in range(0,Jsize):
        for k in range(0,Ksize):
            if dem[j][k] != 0:
                newdem[j][k] = dem[j][k]
    dem=newdem
    np.save(savedirpath+"/"+Date+"/DEM/DEM_Filter_IDW_znum.npy",dem)
    point=[]
    for j in tqdm(range(0,Jsize)):
        for k in range(0,Ksize):
            i= int(dem[j][k])
            if dem[j][k] != 0:
                if voxcel_rgb[i,j,k,0] == 0:
                    r=157
                    g=255
                    b=255
                else:
                    r=voxcel_rgb[i,j,k,0]
                    g=voxcel_rgb[i,j,k,1]
                    b=voxcel_rgb[i,j,k,2]

                arr=[]
                arr.insert(0,(xmin+gridsize*k)+gridsize/2)
                arr.insert(1,(ymax-gridsize*j)-gridsize/2)
                arr.insert(2,(zmin+gridsize*i)+gridsize/2)
                arr.insert(3,r)
                arr.insert(4,g)
                arr.insert(5,b)
                point.append(arr)
    point=np.array(point)
    np.savetxt(savedirpath+"/"+Date+"/DEM/DEM_Filter_IDW_znum_check.txt",point)
```

図 41 IDW メイン処理

このプログラムでは

5.3 Tree_ext.py

Tree_ext のライブラリ一覧を図 42 に示す

```
1  import numpy as np
2  import pandas as pd
3  from tqdm import tqdm
4  import glob
5  import os
6  import cv2
7  import matplotlib.pyplot as plt
```

図 42 Tree_ext のライブラリ一覧

次に、Tree_ext のメイン処理関数を図 43 に示す。

```
def Tree_ext(savedirpath,Date,gridsize):
    Size_voxcel=np.load("Voxcel_size.npy")
    Isize=Size_voxcel[0]
    Jsize=Size_voxcel[1]
    Ksize=Size_voxcel[2]
    Range=np.load("Range.npy")
    view_DSM=np.load("View_DSM.npy")
    print(np.shape(view_DSM))
    xmin,xmax=Range[0][0],Range[0][1]
    ymin,ymax=Range[1][0],Range[1][1]
    zmin,zmax=Range[2][0],Range[2][1]
    voxcel_rgb=np.load(savedirpath+"/"+Date+"/voxel/voxcelrgb.npy")
    dem=np.load(savedirpath+"/"+Date+"/DEM/DEM_znum.npy")
    distance=0.6 #(m)
    high=3
    high=int(round(high/gridsize,0))
    distance=int(round(distance/gridsize,0))
    Tree_ext=np.zeros((Isize,Jsize,Ksize))
    print(np.shape(Tree_ext))
    number=1
    for j in tqdm(range(1,Jsize-1)):
        for k in range(1,Ksize-1):
            D=0
            Arr=[]
            dem_i=int(round(dem[j][k]+distance,0))
            if dem_i != 0:
                serch_j=j
                serch_k=k
                for i in range (dem_i,Isize-1):
                    next_voxcel=np.array(voxcel_rgb[i+1,serch_j-1:serch_j+2,serch_k-1:serch_k+2,0])
                    row_size,col_size=np.shape(next_voxcel)
                    if np.sum(next_voxcel) != 0 and row_size==3 and col_size == 3:
                        D=D+1
                        row_arr=[]
                        col_arr=[]
                        for row in range(0,3):
                            for col in range(0,3):
                                if next_voxcel[row][col] != 0:
                                    arr=[]
                                    row_arr.append(row)
                                    col_arr.append(col)
                                    arr.insert(0,i+1)
                                    arr.insert(1,serch_j-1+row)
                                    arr.insert(2,serch_k-1+col)
                                    Arr.append(arr)
                                    #print(row,col)
                                    #print(next_voxcel)
                                    #print(next_voxcel[row][col])
                                    #print(i+1,serch_j-1+row,serch_k-1+col)
                                    #print(voxcel_D[i+1][serch_j-1+row][serch_k-1+col])
                                    #print("------------------------")
                                    #break
                        next_row=np.array(row_arr)
                        next_col=np.array(col_arr)
                        next_row=int(round(np.mean(next_row),0))
                        next_col=int(round(np.mean(next_col),0))
                        serch_j=next_row-1+serch_j
                        serch_k=next_col-1+serch_k
                    else:
                        break
                if view_DSM[serch_j][serch_k][5]*2/3 <(zmin+gridsize*i):
                    for Len in Arr:
                        if Len[0] <Isize and Len[1] <Jsize and Len[2] <Ksize:
                            tree_I=Len[0]
                            tree_J=Len[1]
                            tree_K=Len[2]
                            Tree_ext[tree_I][tree_J][tree_K]=number
                    number=number+1
    print(number)
    point=[]
    for i in tqdm(range(0,Isize)):
        for j in range(0,Jsize):
            for k in range(0,Ksize):
                if Tree_ext[i][j][k] != 0:
                    arr=[]
                    arr.insert(0,(xmin+gridsize*k)+gridsize/2)
                    arr.insert(1,(ymax-gridsize*j)-gridsize/2)
                    arr.insert(2,(zmin+gridsize*i)+gridsize/2)
                    arr.insert(3,voxcel_rgb[i][j][k][0])
                    arr.insert(4,voxcel_rgb[i][j][k][1])
                    arr.insert(5,voxcel_rgb[i][j][k][2])
                    point.append(arr)
    point=np.array(point)
    os.makedirs(savedirpath+"/"+Date+"/Tree_extract/", exist_ok=True)
    np.savetxt(savedirpath+"/"+Date+"/Tree_extract/Tree_ext"+str(high)+".txt",point)
    np.save(savedirpath+"/"+Date+"/Tree_extract/Tree_ext_voxcel.npy",Tree_ext)
```

図 43 Tree_ext のメイン処理

5.4 diameter.py

Tree_ext のライブラリ一覧を図 44 に示す

```python
import numpy as np
import pandas as pd
from tqdm import tqdm
import glob
import os
import cv2
```

図 44 diameter のライブラリ一覧

次に、Tree_ext のメイン処理関数を図 45 に示す。

```python
def breast_height_diameter(Date,savedirpath,gridsize):
    high=3
    high=int(round(high/gridsize,0))
    Size_voxcel=np.load("Voxcel_size.npy")
    Isize=Size_voxcel[0]
    Jsize=Size_voxcel[1]
    Ksize=Size_voxcel[2]
    Range=np.load("Range.npy")
    xmin,xmax=Range[0][0],Range[0][1]
    ymin,ymax=Range[1][0],Range[1][1]
    zmin,zmax=Range[2][0],Range[2][1]
    Tree_ext_id=np.load(savedirpath+"/"+Date+"/Tree_extract/Tree_ext_id.npy")
    Tree_id_List=pd.read_csv(savedirpath+"/"+Date+"/Tree_extract/id_point.csv",index_col=0)
    distance=1.2##
    I_distance=int(round(distance/gridsize,0))
    N=len(Tree_id_List)
    DEM=np.load(savedirpath+"/"+Date+"/DEM/DEM_Filter_IDW_znum.npy")

    c_gridsize=0.01
    ext_range=10
    c_ext_range=int(round(ext_range*gridsize/c_gridsize,0))
    Judgment_number=np.zeros((Isize,Jsize,Ksize,3))
    #print(np.shape(c_cross_section))

    for num in tqdm(range(0,N)):

        #print(num+1)
        ID=Tree_id_List["ID"][num]
        x=Tree_id_List["x"][num]
        y=Tree_id_List["y"][num]
        #print(ID)

        #仮の中心位置算出-----------------------------------------------------------------------
        ID_extract=np.where(Tree_ext_id == ID,Tree_ext_id,0)
        I_list,J_list,K_list=np.where(Tree_ext_id== ID)
        box_center_K=int(round(K_list.mean(),0))
        box_center_J=int(round(J_list.mean(),0))
        if ext_range<=box_center_K<Ksize-ext_range and ext_range<=box_center_J<Jsize-ext_range:
            i=int(DEM[box_center_J,box_center_K]+I_distance)
            cross_section=ID_extract[i,box_center_J-ext_range:box_center_J+ext_range+1,box_center_K-ext_range:box_center_K+ext_range+1]
            #print(i,box_center_J,box_center_K)

            #中心位置算出-----------------------------------------------------------------------------------
            if np.max(cross_section) != 0:
                J0=box_center_J-ext_range
                K0=box_center_K-ext_range

                J_List,K_List=np.where(cross_section != 0)


                #print(Judgment_number)


                Jg=J_List.mean()
                Kg=K_List.mean()

                Jg=int(round(Jg+J0,0))
                Kg=int(round(Kg+K0,0))

                cross_section=ID_extract[i,Jg-ext_range:Jg+ext_range+1,Kg-ext_range:Kg+ext_range+1]
                #print(cross_section)

                x_c=xmin+Jg*gridsize
                y_c=ymax-Kg*gridsize


                c_J0=(Jg-ext_range)*10
                c_K0=(Kg-ext_range)*10
                #print(J,K)
                for col in range(0,len(K_List)):
                    J=J_List[col]+J0
                    K=K_List[col]+K0
                    #print(J,K)
                    Judgment_number[i,J,K,:]=[c_J0,c_K0,num]
```

```python
    #print(c_s_,Judgment_number[i,J,K])
    c_s_row,c_s_col=np.shape(cross_section)
    c_cross_section=np.zeros((c_s_row*ext_range,c_s_col*ext_range,3,N))
    print(np.shape(c_cross_section))
    dataset=np.load("dataset.npy")
    row,col=np.shape(dataset)
    print("row,col",row,col)
    point_List=glob.glob("/media/owner/NONAME/ForestFloor/kanamine/pointcloud/*")
    CCCCCCC=0
    for path in point_List:
        date=path[52:59]
        print(date)
        datapath=glob.glob("/media/owner/NONAME/ForestFloor/kanamine/pointcloud/"+date+"/"+date+".txt")
        if len(datapath) != 0:
            datapath = datapath[0]


            with open(datapath,'r') as f:
                header = next(f)
                for line in tqdm(f):
                    x =float(line.split()[0])
                    y =float(line.split()[1])
                    z =float(line.split()[2])
                    r =float(line.split()[3])
                    g =float(line.split()[4])
                    b =float(line.split()[5])
                    I=int(round((z-zmin)/gridsize,0))
                    J=int(round((ymax-y)/gridsize,0))
                    K=int(round((x-xmin)/gridsize,0))
                    if r>g:
                        if 0<=I<Isize and 0<=J<Jsize and 0<=K<Ksize:
                            if Judgment_number[I,J,K,2] != 0:
                                #print(Judgment_number[I,J,K,2])
                                c_J0=Judgment_number[I,J,K,0]
                                c_K0=Judgment_number[I,J,K,1]
                                num=int(Judgment_number[I,J,K,2])

                                J=int(round((ymax-y)/c_gridsize,0))
                                K=int(round((x-xmin)/c_gridsize,0))
                                #print(J,K)
                                J=int(J-c_J0)
                                K=int(K-c_K0)
                                #print(J,K)
                                J=int(J-c_J0)
                                K=int(K-c_K0)
                                #print(J,K,2,num,r,g,b)
                                c_cross_section[J,K,2,num]=b/255
                                c_cross_section[J,K,1,num]=g/255
                                c_cross_section[J,K,0,num]=r/255
    os.makedirs(savedirpath+"/"+Date+"/img/", exist_ok=True)
    for num in range(0,len(Tree_id_List)):
        img=c_cross_section[:,:,:,num]
        np.save(savedirpath+"/"+Date+"/img/"+str(num+1)+".npy",img)
        print(num+1)
        plt.imshow(img)
        plt.savefig(savedirpath+"/"+Date+"/img/"+str(num+1)+".png")
        plt.show()
        plt.cla()
```

図 45　diameter のメイン処理関数

5.5 CO2 固定量属性データ

ID	樹高成長量	胸高直径	CO2固定量
0	0.29887597	0.895	0.04007507
1	0.7604336891393579	0.655	0.014266564379743385
2	0.3151815265937884	1.695	0.14947273301332764
3	0.27808217354274367	0.125	0.00077515
4	0.2900873708105867	1.63	0.12823763223808227
5	0.5962226788805872	0.035	0.00011011
6	0.23267510696886473	1.44	0.08344293
7	0.41944023292969645	0.77	0.0465885
8	0.2423751861352267	1.105	0.04303566
9	0.21740347893588688	0.51	0.00993475
10	0.29441925	0.7000000000000001	0.02432223
11	0.3360702176733517	0.03	5.600499044394855e-05
12	0.37868878779465714	0.065	0.00025521
13	0.36471197234460867	0.76	0.036053692631673684
14	0.39437548885423235	0.975	0.05467601
15	0.24035294033747037	1.235	0.06380795
16	0.34578863532875487	1.24	0.09761573
17	0.24231940994314827	1.175	0.05866822
18	0.3019080242844761	0.8150000000000001	0.03619337
19	0.42026833024568194	0.64	0.01973965
20	0.3264446633071019	1.03	0.06410689
21	0.26172549	0.5700000000000001	0.015715328975954068
22	0.2510730059227632	0.625	0.01784834
23	0.21188712088216358	0.59	0.013214756438279074
24	0.22314070929527732	0.635	0.013324938583648471
25	0.4896064151062874	1.12	0.11514804447095472

26	0.1715917064680664	0.005	7.45148029608137e-07
27	0.7365406072470013	0.805	0.024406060104166794
28	0.22960292119804337	1.385	0.07954448
29	0.14225043127070236	0.92	0.018145672146476333
30	0.7004012254883719	1.52	0.11169219522392648
31	2.2799369107176735	1.005	-0.1493985
32	0.42542575241152636	0.9	0.0638829
33	0.3640761311607934	0.17	0.00182108
34	0.19230497902212207	0.92	0.027643124826594256
35	0.3485880103556505	1.56	0.13198093761791485
36	0.30673122644437695	1.065	0.01825909
37	0.29463616	0.715	0.01735637
38	0.2933160633904348	0.05	0.00013287
39	0.3548227945963542	0	0
40	0.27746215669530483	0.58	0.00174774
41	1.7073584378133289	0.635	0.06356274
42	0.46387559136758805	0.690000000000000001	0.039924740353118215
43	0.31542441517639536	0.245	0.00352244
44	0.10800247211769952	0.225	0.00091136
45	1.2792605386246751	1.305	0.11451981386118293
46	0.5860356369497469	0.715	0.027459484927675137
47	0.2009338220003977	0.42	0.00428383
48	0.5116425881915546	0.16	-0.0004486
49	0.9672952163044001	0.745	0.048080257796208836
50	1.2873700169278839	0.07	0.00045909
51	0.2986929006743539	0.58	0.018474170984542293
52	0.8638630661943902	0.2	-0.0042832
53	0.4296273944072449	0.700000000000000001	0.010502103471621227
54	0.9198679548160755	0.86	0.025574304749128418
55	1.6843363754642706	1.815	-0.710278

56	0.2633795255220829	0.17	0.00128155
57	0.35290527	0.665	0.02956927