複数 UAV の協調動作を支援する通信

Communications to Support Collaborative Operation of Multiple UAVs

戸辺 義人 (Yoshito TOBE, Ph. D.)
青山学院大学 教授
(Professor, Aoyama Gakuin University, in Media and Governance)
IEEE　情報処理学会　電子情報通信学会　計測自動制御学会
受賞：情報処理学会 DPS ワークショップ 優秀論文賞 (2000 年度)
著書：センサネットワーク技術-ユビキタス情報環境の構築に向けて, 東京電機大学出版局 (2005 年度)
研究専門分野：情報通信工学

あらまし　UAV (Unmanned Aerial Vehicle) は、手動による遠隔操作または制御プログラミングによる自動操縦ができる無人航空機である。現在、その中の一形態であるマルチコプターは、ドローンとして一般的に知られるようになった。本研究では、複数の UAV を協調させる場合の通信機構を開発した。複数用いる長所は、広域で使えることと、同時に使えることの 2 種類考えられ、本研究では、広域協調センシングを研究課題とし、センシングの網羅性を表現する指標を定義した。広域協調センシングにおいては、複数台の UAV があるときに、1 つの UAV において受信したデータをそのまま次の UAV にデータを送信するのではなく、一度バッファリングして通信可能となったときに受け渡す DTN（Delay Tolerant Network）の考え方を採用することにより、UAV で役割分担をして、効率よく 2 次元平面を網羅する手法を確立した。

１．研究の目的・狙い

UAV は、ドローンとも称されるマルチコプターを含む、手動による遠隔操作または制御プログラミングによる自動操縦ができる無人航空機である。UAV が開発された経緯は、軍事利用の背景から始まる。第二次世界大戦の時代から研究は始まっており、当時は、軍で使用するミサイル発射訓練のための標的、または戦闘機の操縦技術・射撃技術工場のための標的として使用することを目的として開発が進められていた。その後、無線機の小型化や電子機器の機能向上により、写真や映像と取得することが可能になり、地偵察任務に使用されたり、航空機関銃などの武装をし、爆撃や航空戦のための攻撃機として使用されたりした。技術が発展していくに連れて、軍事用の UAV の優位性・利用価値が高まり、全世界でより高性能な UAV の開発が進められ、運用されていくようになった。

UAV の活用は広まりつつあるが、多くの場合、単独での操縦になっている。複数の UAV が自律飛行し、さらに共通の目的のために協調動作することが必要となってくると考えられる。例えば、広域にわたる被災地の状況観測とか、1 つの物体を多方面に同時に観測するといったことが考えられ、本研究では、こうした場合に効率よく複数 UAV が協調できるメカニズムを開発することを目的とする。

２．研究の背景

UAV を通信媒体として用いる研究が世界各国で取り組まれている。UAV による災害時の通信経路維持といったことが検討されている[1]。他にも、これまで複数台ロボットの協調として取り組まれていた研究が、UAV の協調という課題となっている。我が国においては、内閣府 SIP（戦略的イノベーション創造プログラム）インフラ維持・更新・マネジメント技術において、橋梁の保守点検に UAV を用いる研究開発が行われている[2]。

通信技術として、2003 年に K. Fall 氏が提唱した DTN という概念がある。これは、送信側から受信側にデータがただちに届く必要はなく、途中のリンク切断時に数分、数時間、数日待つことを設計の中に組み込むネットワークである。DTN 下におけるルーティングに関する研究も行われている[3]。

３．複数 UAV を用いた広域センシング

ある特定のエリアのデータを取得するエリアセンシングにおいて、エリア全体のデータを広く素早く取得しなければならない状況を想定している。例えば、災害が発生した時など、全体の状況をいち早く把握したい場合などが挙げられる。すべてのデータを取得して

から送信するのでは被害が大きくなってしまうが、全体の状況を素早く把握することができれば、対策を講じやすくなり被害を抑えられる。

そこで本研究では、センシングを完了した状況を把握するために、MSCR（Multi-Scale Coverage Ratio）[4]という指標を定義する。

(1) 図 1 に示すとおり、センシングエリアが長方形の 2 次元平面からなるとする。本平面を縦方向、横方向各々に 2 分割し、合計 4 つの領域に分割する。これを第 1 次層とする。
(2) 本第 1 次層の各領域を一つの平面とみなして、(1) 同様に、4 つの領域に分割して第 2 次層とする。
(3) これを順次繰り返し、第 n 次層を定義する。
(4) UAV により、ある点の情報が取得済みであるということは、複数の層全体でのカバーを考えて、MSCR を定義する。
(5) MSCR m は、

$$m = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{4^i}\sum_{j=1}^{4^i} q_{i,j(i)}$$

で与えられる。ただし、ここに qi, j は、第 i 次層において j 番目のスロットのセンシングが完了していれば 1、完了していなければ 0 の値を取る変数である。

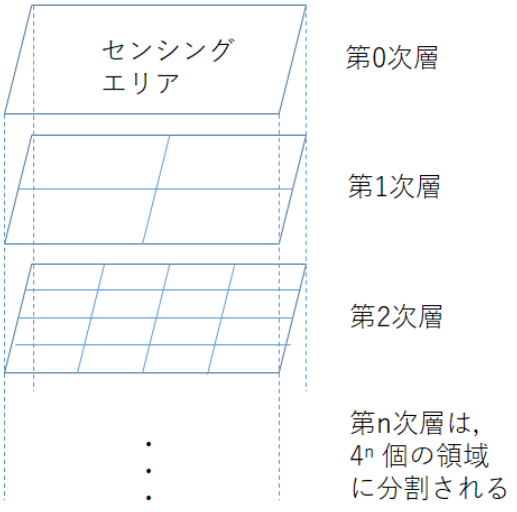

図 1　MSCR を定義する際の階層

4．UAV ノード動作およびシミュレーション

本章では、MSCR を考慮して UAV のノードを定義し、各種動作パターンのシミュレーション結果を示す。

4．1　想定シナリオ

複数台の UAV が各々のセンシングしたデータをデータ収集局に送信するものとする。この際、センシング動作を行う UAV を SensorNode と GatewayNode に分類する。

- SensorNode：無線通信距離が短く、GatewayNode へセンシングデータを送ることで最終的にデータ収集局へデータが届くようにする。
- GatewayNode：データ収集局との通信が可能な大きい無線通信距離を有し、SensorNode からデータを受信する。GatewayNode と SensorNode が互いに通信可能距離 r 以内にいるときに限り、GatewayNode は SensorNode からデータを受け取ることとする。

実際には、SensorNode が GatewayNode と同程度の機能を持ってセンシングする場合や、複数台の GatewayNode が行動する場合など、より多くのパターンが想定されるが、本研究では、単一の UAV は SensorNode か GetewayNode のいずれかであるとする。GatewayNode と SensorNode の動き方は、特定のポリシーによって決定する。

[センシング動作の手順]
(1) センシングタスク指示を受けた GatewayNode と SensorNode は、エリアに対して飛び立つ。
(2) 各 SensorNode は、開始位置に到着後、センシングを開始する。
(3) センシング開始後、各ノードは与えられたセンシングポイントへ向かって移動する。センシングポイントに到達したら、センサでデータ取得を開始する。
(4) 移動、センシングが終わった際には、通信可能距離範囲内に GatewayNode がいるか確認する。
(5) 通信可能距離範囲内にいれば、SensorNode と GatewayNode の通信を開始し、データの送受信を行う。

複数 UAV の協調動作を支援する通信

Communications to Support Collaborative Operation of Multiple UAVs

(6) データを受け取った GatewayNode は、データ収集局にデータを送信する。

[接続方式]

GatewayNode と SensorNode の通信可能距離とセンシングエリア全体の大きさに応じて、次の 3 種類の接続方式が考えらる。

(1) 完全接続方式
(2) 不完全接続方式 GatewayNode 固定型
(3) 不完全接続方式 GatewayNode 移動型

完全接続方式では、SensorNode がどの位置にいても GatewayNode と通信可能であり、GatewayNode は固定位置にいればよい。不完全接続方式は、SensorNode と GatewayNode が直接通信できない場合である。

・不完全接続方式 GatewayNode 固定型

SensorNode はセンシングしながら移動を続け、GatewayNode はある一定の場所から動かない。したがって、SensorNode はデータが一定数集まれば、データを GatewayNode に渡すために通信可能距離 r に入るように移動する必要がある。r に入って通信可能状態になったら、データの受け渡しを行う（図 2）。

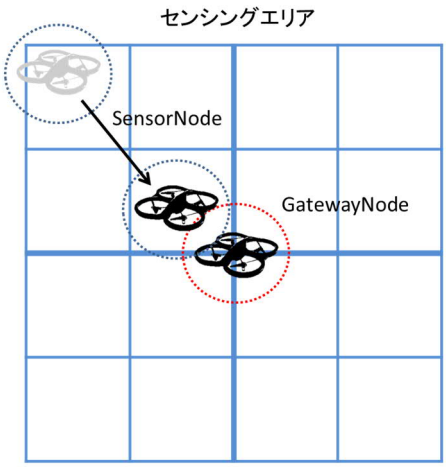

図 2　GatewayNode 固定型

・不完全接続方式 GatewayNode 移動型

SensorNode はセンシングしながら移動を続け、GatewayNode は各エリアを周回するように移動し、各 SensorNode からデータを回収していく。このとき、SensorNode は自らデータを届けに行く必要はなく、周回している GatewayNode が自分の担当エリアに来て r 以内に入り、通信可能状態となった時にデータを送信すればよい（図 3）。

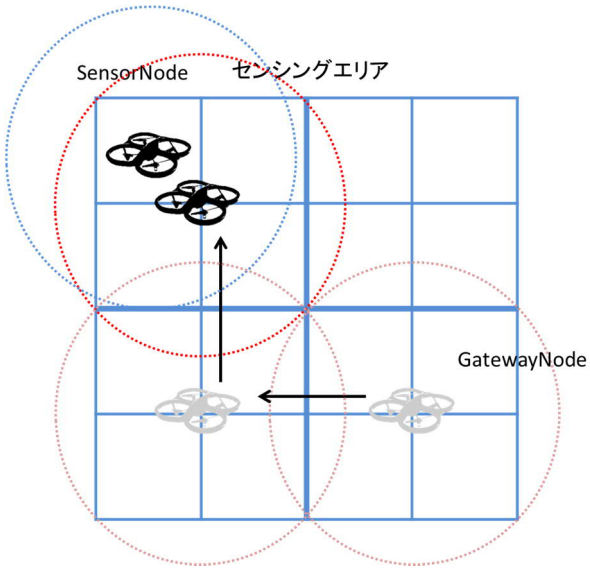

図 3　GatewayNode 移動型

[センシング移動パターン]

SensorNode は、あらかじめ決められた地点をセンシングするが、その移動ルートは異なる。ここでは、BFS（幅優先探索：Breadth first search）*1型、xy-sweep 型、Eddy-sweep 型の 3 種類の移動方式を考える。

・BFS 型

SensorNode は、担当エリアを BFS のアルゴリズムに従って移動していく。はじめは全体をおおまかに網羅するように移動し、次第に細かく移動していく（図 4）。

・xy-sweep 型

SensorNode は、担当エリアを x 軸、y 軸のいずれかに沿ってセンシングしながら移動し、もう一方の軸方向にずらして動作を繰り返す（図 5）。

・Eddy-sweep 型

SensorNode は、担当エリアを外側から内側へ、中心に向かって渦を描くように移動していく（図 6）。

複数 UAV の協調動作を支援する通信

Communications to Support Collaborative Operation of Multiple UAVs

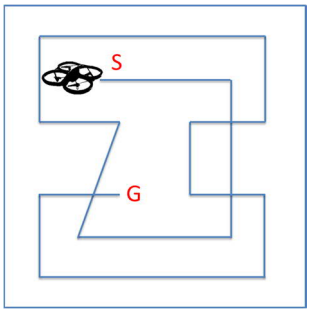

図４ａ　BFS 移動パターン

```
 depthMax = environment.calcDepth
depth = 1
offset_x = 0
offset_y = 0
sensingWidth = environment.getSensingWidth
sensingHeight = environment.getSensingHeight
order = 0

breathFirstSearch(depthMax, depth, offset_x, offset_y,
                  sensingWidth, sensingHeight, order)
 result = []
 if depthMax < depth
  return

 depth += 1
 result.add( d - 1, offset_x + (sensingWidth / 2),
          offset_y + (sensingHeight / 2), order)
 result.add(breathFirstSearch(depthMax, d, offset_x,
          offset_y, sensingWidth / 2, sensingHeight / 2, 1)
 result.add(breathFirstSearch(depthMax, d, offset_x + (sensingWidth / 2),
          offset_y, sensingWidth / 2, sensingHeight / 2, 2)
 result.add(breathFirstSearch(depthMax, d, offset_x + (sensingWidth / 2),
          offset_y + (sensingHeight / 2), sensingWidth / 2, sensingHeight / 2, 3)
 result.add(breathFirstSearch(depthMax, d, offset_x, offset_y + (sensingHeight
/ 2),
          sensingWidth / 2, sensingHeight / 2, 4)
end breathFirstSearch
```

図４ｂ　BFS の擬似コードによる表現

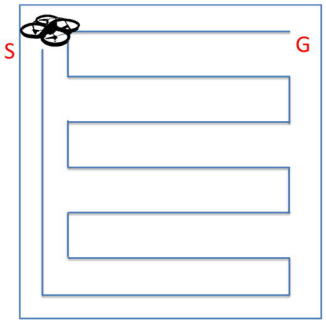

図５ａ　xy-sweep 移動パターン

```
sensingRadius = uav.getSensingRadius
sensingWidth = environment.getSensingWidth
sensingHeight = environment.getSensingHeight

result = []
count_x = sensingWidth / (sensingRadius * 2)
if (sensingWidth / (sensingRadius * 2)) - count_x > 0
 count_x += 1
intervalWidth = sensingWidth / count_x

count_y = sensingHeight / (sensingRadius * 2)
if (sensingHeight / (sensingRadius * 2)) - count_y > 0
 count_y += 1
intervalHeight = sensingHeight / count_y

seq = 0
if uav.direction == VERTICAL
 for i = 0 to count_x
  temp = []
  for j = 0 to count_y
   seq += 1
   temp.add(seq, quota_x + (intervalWidth / 2) + intervalWidth * i,
         quota_y + (intervalHeight / 2) + intervalHeight * j)
  end for

  if i % 2 == 0
   for k = 0 to (temp.size - 1)
    result.add(temp[k])
    k = k + 1
   end for
  end if

  if i % 2 == 1
   for k = (temp.size - 1) to 0
    result.add(temp[k])
    k = k - 1
   end for
  end if
 end for

elsif uav.direction == HORIZONTAL
 for i = 0 to count_y
  temp = []
  for j = 0 to count_x
   seq += 1
   temp.add(seq, quota_x + (intervalWidth / 2) + intervalWidth * j,
         quota_y + (intervalHeight / 2) + intervalHeight * i)
  end for

  if i % 2 == 0
   for k = 0 to (temp.size - 1)
    result.add(temp[k])
    k = k + 1
   end for
  end if
  if i % 2 == 1
   for k = (temp.size - 1) to 0
    result.add(temp[k])
    k = k - 1
   end for
  end if

 end for
end elseif
```

図５ｂ　xy-sweep の擬似コードによる表現

複数 UAV の協調動作を支援する通信

Communications to Support Collaborative Operation of Multiple UAVs

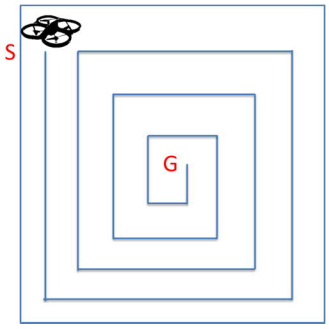

図６ａ　Eddy-sweep 移動パターン

```
sensingRadius = uav.getSensingRadius
sensingWidth = environment.getSensingWidth
sensingHeight = environment.getSensingHeight
quota_x = uav.getQuotaX
quota_y = uav.getQuotaY

count_x, count_y, intervalWidth, intervalHeight
 = calc_interval(sensingRadius, sensingWidth, sensingHeight)

result = []
seq = 0
previous_point_x = quota_x + (intervalWidth / 2)
previous_point_y = quota_y + (intervalHeight / 2)
result.add(seq, previous_point_x, previous_point_y)

rotate = 0
if uav.direction == CLOCKWISE
 for i = 1 to count_x
  seq += 1
  result.add(seq, previous_point_x += intervalWidth * Math.sin(rotate *
Math.PI / 2),
          previous_point_y += intervalHeight * Math.cos(rotate * Math.PI / 2))
 end for

 rotate += 1
 until count_x == 0 && count_y == 0 do
  for j = 1 to count_y
   seq += 1
   result.add(seq, previous_point_x += intervalWidth * Math.sin(rotate *
Math.PI / 2),
           previous_point_y += intervalHeight * Math.cos(rotate * Math.PI / 2))
  end for
  count_y -= 1
  rotate += 1

  for k = 1 to count_x
   seq += 1
   result.add(seq, previous_point_x += intervalWidth * Math.sin(rotate *
Math.PI / 2),
           previous_point_y += intervalHeight * Math.cos(rotate * Math.PI / 2))
  end for
  count_x -= 1
  rotate += 1

 end until
end if
```

図６ｂ　Eddy-sweep の擬似コードによる表現

４．２ シミュレーション

SensorNode と GatewayNode の接続方式３種類と SensorNode の移動パターン 3 種類の組み合わせから、合計 9 種類の動作が考えられるが、ここでは不完全接続 GatewayNode 移動型について、３つの移動パターンが MSCR の変化に与える影響をシミュレーションにより検証する。

[シミュレーションの条件]

表 1、表 2 の条件を想定し、UAV の台数として

- GatewayNode　1 台
- SensorNode　4 台

を仮定する。

表１　センシングエリアエリアの設定

要素	値	単位
エリアの縦幅	2,000	m
エリアの横幅	2,000	m
SensorNode 1 台あたりの担当エリアの縦幅	1,000	m
SensorNode 1 台あたりの担当エリアの横幅	1,000	m
単位時間 TIME_TICK	0.1	ms

表２　仮定する UAV の仕様

要素		値	単位
UAV 同士の通信可能距離	完全接続型	100	m
	不完全接続型	100	m
電力最大量		1000	mW
消費電力量	移動	0.06	mW/m
	センシング	0.05	mW/回
センシングブロック 1 箇所あたりの縦幅		50	m
センシングブロック 1 箇所あたりの横幅		50	m

複数 UAV の協調動作を支援する通信

Communications to Support Collaborative Operation of Multiple UAVs

本条件において、BFS、xy-sweep、Eddy-Sweep により、MSCR の変化を示したものが図 7 となる。この結果、この 3 種類の移動パターンの中では、BFS が MSCR 向上に適していることがわかった。このシミュレーション結果は一例であり、さまざまな移動パターン、パラメータにより、カバー効率のよい移動を決定するシミュレータを本研究で完成することができた。

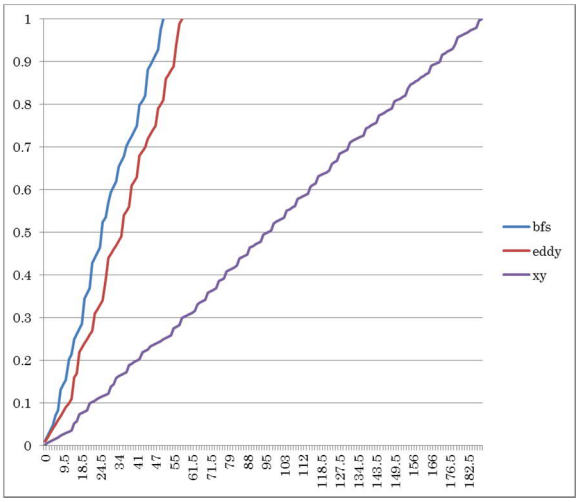

図 7　BFS, xy-sweep, Eddy-sweep vs. MSCR

５．将来展望

3 次元空間を自由に行き来できる UAV を監視･観測に用いる可能性は、今後広がっていく。風の影響、バッテリ持続時間といったハードウェア上の未解決問題もあるが、そうした問題が解決された将来、複数の UAV を連携させるアプリケーションは多く出てくる。そのときに、飛行性能と無線到達距離の両方のバランスを考慮した協調方式が必要になってくる。さらに、地上の機器との連携も考えていく。

参考文献

[1] M. Erdelj, E. Natalizio, K. R. Chowdhury and I. F. Akyildiz, "Help from the Sky: leveraging UAVs for disaster management", IEEE Pervasive Computing: 16(1), pp. 24-32, 2017.

[2] 内閣府 SIP インフラの維持点検 http://www.jst.go.jp/sip/k07.html 戦略的イノベーション創造プログラム: インフラ維持管理・更新・マネジメント技術

[3] K. Fall, "A delay-tolerant network architecture for challenged Internet", ACM SIGCOMM '03, pp. 27-34, 2003.

[4] T. Watanabe, N. Thepvilojanapong, Y. Ohnogi, Y. Ohta, J. Takahashi and Y. Tobe, "Cooperative control of multiple UAVs based on hierarchical coverage ratio", Int. Workshop on Smart Sensing Systems, 2016.

用語解説

*1 BFS（Breadth First Search）

グラフ理論における木において、根から探索を始める場合に、同じ深さにある方向を先に探索する方法。もう一方の方法は、DFS（Depth First Search）である。

この研究は、平成26年度SCAT研究助成の対象として採用され、平成27～28年度に実施されたものです。