**고급소프트웨어실습**

**중간 리포트**

**(CSE 4152)**

**Due: 2021년, 10/20일, 23:59분**

**학번 : 20171646**

**반 번호 : 4반**

**이름 : 박태윤**

**1. 1주 차에서 다룬 두 가지 예제 외에 몬테카를로 시뮬레이션을 사용하여 해결할 수 있는 문제를 2가지 이상 들고 어떻게 몬테카를로 시뮬레이션을 적용할 수 있는지 설명하시오. (1페이지 이내)**

(삼차원 공간에서 물체의 부피 계산)

부피를 계산하고자 하는 영역이 여러 도형의 겹치는 부분 등 부피 계산이 복잡할 경우에 몬테 카를로 시뮬레이션을 이용하여 이에 대한 부피 값을 구할 수 있다. 특정 물체의 영역을 모두 포함하는 직육면체 또는 정육면체의 범위 내에서 난수 순서쌍을 만들어 시뮬레이션을 하고자 하는 횟수인 K번 만큼 전체 맵에 맵핑을 하고 부피를 구하고자 하는 물체를 기준으로 해당 물체의 안쪽과 그 바깥의 영역을 구분을 한다. 삼차원 공간에서는 전체 공간을 부피를 계산하고자 하는 물체의 전체 영역을 포함하는 부피 계산이 다소 용이한 직육면체 혹은 정육면체로 표현을 할 수 있고, 전체 맵핑을 한 횟수 K에 대해 물체 안쪽에 맵핑이 된 점들의 개수의 퍼센테이지 값을 구해 계산을 한 전체 영역의 부피에 이 퍼센테이지를 곱하게 되면 물체 부피에 대한 근사치를 구할 수 있게 된다.

(몬테카를로 트리 탐색)

몬테카를로 트리 탐색은 특히 게임에서 어떤 동작으로 움직이는 것이 가장 적절한 것인지를 트리 탐색을 통해 결정하는 방식이다. 이는 탐색 트리의 모든 노드를 탐색하지 않고, 시뮬레이션을 통해 게임에서 승리 가능성이 높아 보이는 방향으로 탐색이 진행된다. 트리의 각 노드에는 게임 실행 횟수와 승리 횟수, 동작이 주어져 있는데, 자식 노드들의 승리 횟수의 전체 합이 부모의 승리 횟수이고 마찬가지로 자식 노드들의 총 실행 횟수의 합이 부모의 실행 횟수로 트리 노드의 요소가 구성이 된다. 트리를 탐색할 때 현재 노드의 정보에서 높은 승률을 가지는 자식 노드를 우선으로 탐색하면서 해당 노드의 동작을 진행하고, 아직 expansion이 되지 않은 깊이까지 내려갔을 경우 이 때 게임이 끝날 때까지 가능한 동작에 대해 random한 선택으로 시뮬레이션을 게임이 끝날 때까지 진행하여 승리 혹은 패배의 결과를 도출한 후 이 결과를 다시 연결 되어있는 부모 및 조상 노드에 backpropagation을 하는 방식이다. 즉, 시뮬레이션을 하였을 때 최종적으로 lose하는 결과를 얻는다면 연결된 조상 노드들은 실행 횟수만 1이 증가되고 만약 win하였다면 실행 횟수와 승리 횟수가 모두 1씩 증가하는 방식이다. 이러한 방식으로 단순 시뮬레이션을 통해 실제로 카운트한 실행 횟수 대비 승리 비율이 가장 높은 방향을 채택하여 게임에서 어떠한 방향으로 동작을 하는 것이 가장 승률이 높은 선택인지를 채택할 수 있다.

**2. Pearson Correlation Coefficient와 Principal Component Analysis (PCA)는 유사한 목적으로 사용될 수 있다. 어떤 경우에 두 가지 방법이 유사한 목적으로 사용될 수 있는지 구체적인 예를 들어서 설명하시오. (1 페이지 이내)**

pearson correleation coefficient는 변수 간의 상관 관계를 수치적으로 표현하고자 할 때 사용이 된다. +1은 양의 상관관계, 0에 가까울수록 상관관계가 없다고 볼 수 있고 -1은 음의 선형 상관관계를 나타낸다고 할 수 있다. pca는 고차원의 데이터를 가지고 공분산 행렬을 구해 여기서 고유값과 그에 대응되는 고유벡터들을 찾고 고유값 기준으로 정렬이 된 고유벡터 중에서 고유값이 큰 일부만을 선택을 해 데이터의 차원을 줄일 수 있는 방법이다. 공분산행렬에서 고유 값이 큰 고유벡터들, 즉 중요한 feature들을 선택하는 pca의 방식을 pearson correleation coefficient에서는 상관관계가 높은 순대로 variable간의 관계를 채택하는 방식으로 PCA와 유사한 역할을 수행할 수 있다.

두 알고리즘 모두 얼굴 인식이라는 목적으로 사용될 수 있다. 얼굴 인식을 하고자 할 때, 이미지는 픽셀로 표현이 되고 픽셀 값을 연결하여 픽셀 개수만큼의 차원 벡터를 만들고 여기서 pca를 수행하여 주성분 벡터를 고유 값 순으로, 즉 중요도 순으로 특정 값인 K개로 고른 후에 차원 축소를 진행한 이미지를 통해 얼굴 인식을 할 수 있다. 이렇게 차원 축소를 한 이미지는 상대적으로 중요한 정보만을 담고 있는 이미지 이기 때문에 차원 축소를 하기 전 모든 정보를 다 따져가며 정보를 비교하는 것 보다 훨씬 용이하다고 할 수 있다. 왜냐하면 중요도 없이 모든 정보를 비교해가며 차이를 따지면 계산 복잡도도 늘어날 뿐 아니라 얼굴 인식에 있어서 상대적으로 중요하지 않은 정보의 차이 때문에 얼굴 인식에 실패할 가능성이 생길 수 있기 때문이다. 따라서 pca를 이용하여 중요도 순으로 고유 벡터를 골라 이에 대해서 비교를 진행한다면 훨씬 좋은 얼굴 인식 비교 결과를 얻을 수 있다. 피어슨 상관 계수를 통해서도 얼굴 인식을 할 수 있을 것이다. 예를 들어 이미지 상에서 나타나는 눈의 간격, 콧볼의 크기 등을 값으로 계산을 하여 이들 간의 상관 관계를 pearson correleation coefficient 방법을 통해 계산을 하고 상관 관계가 가장 큰, 즉 -1 ~ +1의 값을 가지는 수 중 절대값이 큰 정보만을 비교하여 얼굴 인식을 진행할 수 있다. 상관 계수의 절대 값이 낮은 정보들 간의 관계는 따로 얼굴 인식 과정에서 보지 않고 현재 얼굴 인식을 하고자 하는 사람의 얼굴이 기존에 있던 이미지 정보로 계산이 된 피어슨 상관 계수 값에서 절대값이 큰 정보만을 비교하는, 마치 앞선 pca에서 상대적으로 얼굴 인식에서 중요한 정보만을 가지고 비교하는 방식과 유사하게 얼굴 인식을 시도하고 이에 따른 결과를 얻을 수 있다.

**3.** [**https://www.math.utah.edu/software/c-with-fortran.html**](https://www.math.utah.edu/software/c-with-fortran.html) **문서를 읽고 C/C++ 언어와 FORTRAN 언어의 차이를 요약한 후, C/C++ 프로그램에서 FORTRAN 함수를 호출할 때 주의할 점을 기술하시오. (1 페이지 이내)**

C/C++와 FORTRAN은 우선 데이터 유형의 차이가 존재한다. 기본적으로 정수, 실수, 문자형 등의 두 언어에서 공통적으로 존재하는 데이터 타입이 있지만, FORTRAN에서는 C/C++에 없는 복소수형 데이터 타입이 존재한다. 이는 COMPLEX로 표현이 된다. 함수 반환 type에서도 차이를 보이는데, 배열을 반환할 수 없는 등의 공통점이 존재하는 반면 Fortran에서는 오로지 scalar값만을 반환할 수 있고 C/C++에서는 해당 언어의 구조체나 C++에서는 class를 반환할 수 있다는 차이 또한 존재한다. 그리고 배열의 인덱싱 순서가 다르다는 차이도 존재하는데, C/C++는 기본적으로 row-major-order를 따르지만 Fortran에서는 이와는 반대로 column-major-order를 따른다는 차이점이 존재한다. 즉, C/C++에서는 인덱싱에서 (1,1) -> (1,2) -> (1,3)….과 같은 순서를 따르는 반면에 Fortran에서는 (1,1) -> (2,1) -> (3,1)과 같은 순서를 따른다는 것이다. 이외에 함수에서 매개변수를 호출하는 방식에 있어서 C/C++은 call by value방식을 따르지만 fortran은 call by reference방식을 따른다는 차이점 또한 존재한다.

이러한 차이점 때문에 C/C++에서 fortran함수를 호출할 때 몇 가지 처리를 해줘야 하는 부분이 생긴다. 첫번째로, 앞서 언급한 fortran에서는 함수의 인자를 값이 아닌 주소로 넘기는 call by reference방식을 사용하기 때문에 만약 C/C++에서 fortran의 함수를 호출할 때 단순하게 해당 언어에서 함수를 호출하는 것처럼 인자를 넘긴다면 오류가 발생할 수 있기 때문에 C/C++에서 주소를 표현하는 포인터로 인자를 넘겨주어야 적절한 사용을 할 수 있다. 또한 배열 저장 방식에 있어서 row-major와 column-major의 차이가 존재하기 때문에 C/C++에서 그대로 배열을 인자로 넘겨준다면 배열을 사용하는 방식에 있어서 잘못된 참조가 생길 수 있기 때문에 행렬을 transpose한 결과를 인자로 넘겨주어 fortran함수를 호출하거나 반대로 fortran함수를 이용하여 계산 결과를 받았다면 C/C++에서는 이를 다시 transpose하여 사용하는 등의 처리를 따로 해주어야 한다.