High Performance Parallel Programming (CS61064)

Week – 4
Part 1

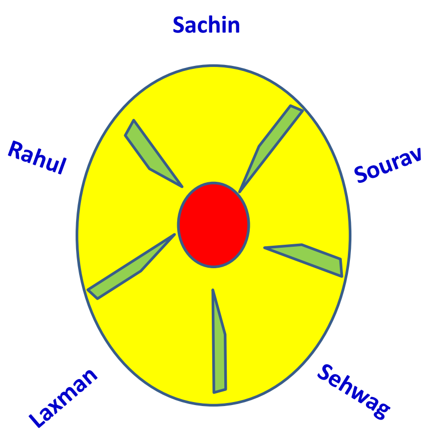
Pralay Mitra

Lock Functions

- omp_init_lock
- omp_destroy_lock
- omp_set_lock
- omp_unset_lock
- omp_test_lock

- omp_init_nest_lock
- omp_destroy_nest_lock
- omp_set_nest_lock
- omp_unset_nest_lock
- omp_test_nest_lock

Deadlocks in OpenMP



Dining Philosophers Problem

Deadlocks in OpenMP

```
worker ()
{
   #pragma omp barrier
}
main ()
{
   #pragma omp parallel sections
   {
       #pragma omp section
           worker();
   }
}
```

- The fork is a semaphore.
- You need two forks to eat, but you have to get one at a time
- If everybody gets one, and just wait for the other …. nobody will eat.

Race condition in openMP

```
#pragma omp parallel shared(b) private(errors)
{
  #pragma omp for nowait
        for(i = 0; i < 10; i++)
        dt[i] = b + dt[i]*5;
    errors = dt[9] + 1;
}
```

Ma, Hongyi, et al. "Symbolic
analysis of concurrency errors in

Combined Parallel Work-sharing Constructs

- **parallel sections Construct**

#pragma omp parallel sections *[clause[[,] clause] ...] new-line*
{
*[***#pragma omp section** *new-line]*
structured-block
*[***#pragma omp section** *new-line*
structured-block]

...
}

Deadlock in openMP

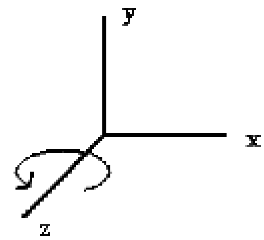
```
void print_results(float array[N], int section) {
        #pragma omp critical {
                int tid = omp_get_thread_num();
                printf("The results are in section %d.\n", section);
                for (i = 0; i < N; i++)
                        printf("%e ",array[i]);
        } /*end of critical*/
        #pragma omp barrier
        printf("Thread %d is done.\n", tid);
}
#pragma omp sections {
        #pragma omp section
                print_results(c, 1);
        #pragma omp section
                print_results(c, 2);
} /*end of parallel section*/
```

Ma, Hongyi, et al. "Symbolic
analysis of concurrency errors in

Problem 1

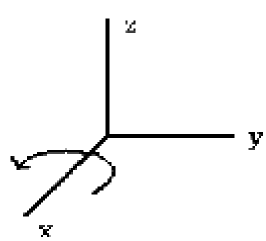
Rotation About an Arbitrary Axis in 3D

Rotation About Z-Axis in 3D




```
x' = x*cos q - y*sin q
y' = x*sin q + y*cos q
z' = z
          ( cos q   -sin q    0    0)
Rz (q) =  ( sin q    cos q    0    0)
          ( 0        0        1    0)
          ( 0        0        0    1)
```

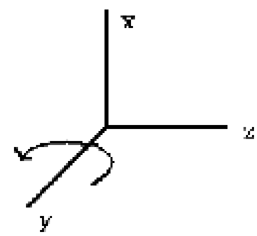
Rotation About X-Axis in 3D



y' = y*cos q - z*sin q
z' = y*sin q + z*cos q
x' = x

$$Rx(q) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos q & -\sin q & 0 \\ 0 & \sin q & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation About Y-Axis in 3D




```
z' = z*cos q - x*sin q
x' = z*sin q + x*cos q
y' = y

         (cos q       0       sin q     0)
Ry(q) =  (0           1         0       0)
         (-sin q      0       cos q     0)
         (0           0         0       1)
```

Rotation About an Arbitrary Axis in 3D

(1) Translate space so that the rotation axis passes through the origin.

(2) Rotate space about the z axis so that the rotation axis lies in the xz plane.

(3) Rotate space about the y axis so that the rotation axis lies along the z axis.

(4) Perform the desired rotation by θ about the z axis.

(5) Apply the inverse of step (3).

(6) Apply the inverse of step (2).

(7) Apply the inverse of step (1).

Rotation About an Arbitrary Axis in 3D

The matrices for rotation by α around the x-axis, β around the y-axis, and γ around the z-axis

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation About an Arbitrary Axis in 3D

The general rotation matrix depends on the order of rotations. The first matrix rotates about x, then y, then z; the second rotates about z, then y, then x.

$$R_z R_y R_x = \begin{bmatrix} \cos\beta\cos\gamma & \cos\gamma\sin\alpha\sin\beta - \cos\alpha\sin\gamma & \cos\alpha\cos\gamma\sin\beta + \sin\alpha\sin\gamma & 0 \\ \cos\beta\sin\gamma & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & -\cos\gamma\sin\alpha + \cos\alpha\sin\beta\sin\gamma & 0 \\ -\sin\beta & \cos\beta\sin\alpha & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_x R_y R_z = \begin{bmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta & 0 \\ \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\beta & 0 \\ \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Limitations

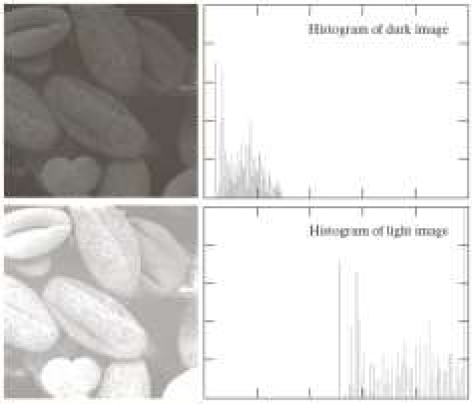
- Computationally slow
- Not recommended for large scale application
- Alternative is Quaternion based method.

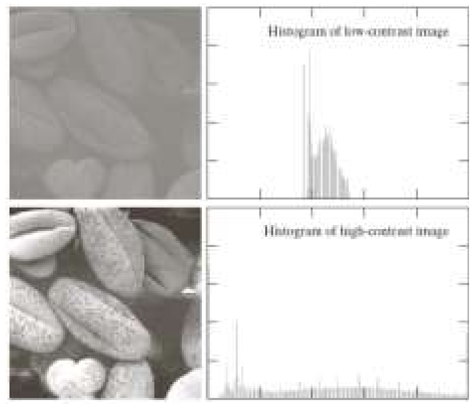
$$\begin{bmatrix} c + a_x^2(1-c) & a_x a_y(1-c) - a_z s & a_x a_z(1-c) + a_y s \\ a_y a_x(1-c) + a_z s & c + a_y^2(1-c) & a_y a_z(1-c) - a_x s \\ a_z a_x(1-c) - a_y s & a_z a_y(1-c) + a_x s & c + a_z^2(1-c) \end{bmatrix}$$

Problem 2

Histogram Equalization

Application in Image Processing





Application in Image Processing

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

Intensity distribution and histogram values for a 3-bit, 64×64 digital image.

Application in Image Processing

- Histogram Equalization – Example

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7\,p_r(r_0) = 1.33\ [1]$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7\,p_r(r_0) + 7\,p_r(r_1) = 3.08\ [3]$$

$$s_2 = 4.55\ [4],\ s_3 = 5.67\ [5],\ s_4 = 6.23\ [6],$$

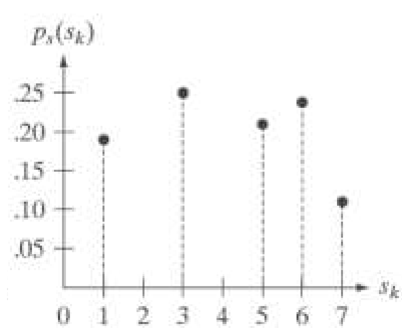
$$s_5 = 6.65\ [7],\ s_6 = 6.86\ [7],\ s_7 = 7.00\ [7]$$

Application in Image Processing

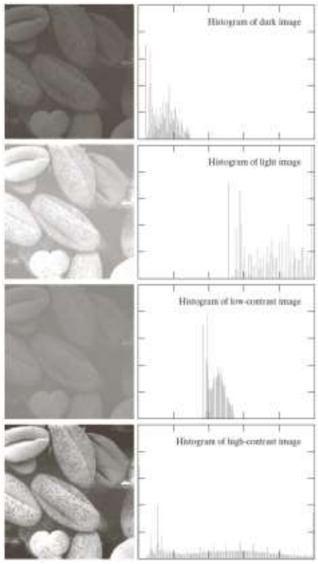
- Histogram Equalization – Example

$S_0 = 1.33 \rightarrow 1$
$S_1 = 3.08 \rightarrow 3$
$S_2 = 4.55 \rightarrow 5$
$S_3 = 5.67 \rightarrow 6$
$S_4 = 6.23 \rightarrow 6$
$S_5 = 6.65 \rightarrow 7$
$S_6 = 6.86 \rightarrow 7$
$S_7 = 7.00 \rightarrow 7$

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02



Application in Image Processing



Original Image

Workout

Histogram equalized Image

Problem 3

Gaussian Elimination

Gaussian Elimination

- Gaussian elimination aims to transform a system of linear equations into an upper-triangular matrix in order to solve the unknowns and derive a solution. A pivot column is used to reduce the rows before it; then after the transformation, back-substitution is applied.

System of equations	Row operations	Augmented matrix
$\begin{aligned} 2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{aligned}$	**Workout**	$\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$
$\begin{aligned} 2x + y - z &= 8 \\ \tfrac{1}{2}y + \tfrac{1}{2}z &= 1 \\ 2y + z &= 5 \end{aligned}$	$L_2 + \tfrac{3}{2}L_1 \to L_2$ $L_3 + L_1 \to L_3$	$\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ 0 & \tfrac{1}{2} & \tfrac{1}{2} & 1 \\ 0 & 2 & 1 & 5 \end{array}\right]$
$\begin{aligned} 2x + y - z &= 8 \\ \tfrac{1}{2}y + \tfrac{1}{2}z &= 1 \\ -z &= 1 \end{aligned}$	$L_3 + -4L_2 \to L_3$	$\left[\begin{array}{ccc\|c} 2 & 1 & -1 & 8 \\ 0 & \tfrac{1}{2} & \tfrac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$

Gaussian Elimination

Table 1.
CPU time (seconds) with n = 400 and p = 4

Chunk	default	1	2	4	8	16	32	64	128
Static	0.74	1.46	1.81	1.77	1.15	0.82	0.77	0.66	0.57
Dynamic	2.27	2.53	2.38	2.11	1.41	0.97	0.76	0.61	0.56
Guided	0.78	0.80	0.78	0.81	0.74	0.69	0.68	0.68	0.59

Table 2.
CPU times (seconds) with n = 800 and p = 4

Chunk	default	1	2	4	8	16	32	64	128
Static	8.35	20.89	21.66	21.41	17.50	11.48	10.27	9.47	10.27
Dynamic	22.63	22.54	22.10	28.59	19.21	11.66	9.59	9.74	10.39
Guided	9.33	9.53	9.28	9.47	9.49	9.10	8.95	9.84	11.10

Table 3.
CPU times (seconds) with n = 1200 and p = 4

Chunk	default	1	2	4	8	16	32	64	128
Static	51.01	65.69	66.34	65.57	63.01	56.26	54.88	53.61	53.06
Dynamic	85.38	85.54	85.46	82.27	69.88	51.45	42.54	42.09	43.65
Guided	46.10	46.55	46.24	45.71	45.25	44.58	43.61	43.50	43.24

Table 4.
Load Balancing Speedup results using varying values of n

	400	800	1200
Static	3.95	4.59	2.84
Dynamic	4.02	4.00	3.44
Guided	3.81	4.28	3.35

S.F.McGinn & R.E.Shaw in the Proc. of the 16th Annual International Symp on High Performance Computing Systems and Applications (HPCS'02)

Problem 4

Motif Search